|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Главный инженер ООО «Компания «Стальэнерго»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю. А. Федоркин  «   »          2020 г. |

|  |  |  |
| --- | --- | --- |
| **ОБЪЕКТНЫЙ КОНТРОЛЛЕР ПРИВОДА СТРЕЛКИ**  **ОКПС-Е-К**  ЕИУС.хххххх.ххх 01 91 01 01 | | |
|  | Начальник отдела разработок  новых видов продукции ООО «Компания «Стальэнерго»  \_\_\_\_\_\_\_\_\_\_\_\_\_ И. В. Солодовник  «   »          2020 г. | |

**История изменений**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| п/п | Дата | Номер версии | Статус | Автор | Причина изменения, № извещения | Комментарии |
| 1 |  | 1.0 | Проект |  |  | Создание документа |
|  |  |  |  |  |  |  |

**Содержание**

[1 Введение 5](#_Toc40688938)

[1.1 Назначение документа 5](#_Toc40688939)

[1.2 Термины, определения и сокращения 5](#_Toc40688940)

[1.3 Рекомендации к ознакомлению 5](#_Toc40688941)

[1.4 Ссылки 6](#_Toc40688942)

[2 Аппаратные средства 7](#_Toc40688943)

[3 Внешние интерфейсы 8](#_Toc40688944)

[4 Логическая структура ПО 9](#_Toc40688945)

[4.1 Построение архитектуры 9](#_Toc40688946)

[4.1.1 Принципы обеспечения безопасности прибора 9](#_Toc40688947)

[4.1.2 Принципы построения архитектуры 10](#_Toc40688948)

[4.2 Потоки управления 11](#_Toc40688949)

[4.2.1 Главный поток 12](#_Toc40688950)

[4.2.2 Поток временно́й синхронизации 14](#_Toc40688951)

[4.2.3 Поток синхронизации данных 15](#_Toc40688952)

[4.2.4 Поток обработки неиспользуемых прерываний 16](#_Toc40688953)

[4.3 Обмен данными между потоками 16](#_Toc40688954)

[4.4 Синхронизация данных между процессорами 19](#_Toc40688955)

[4.4.1 Синхронизация данных 19](#_Toc40688956)

[4.4.2 Синхронизируемые параметры 19](#_Toc40688957)

[4.4.3 Идентификатор синхронизируемого параметра 20](#_Toc40688958)

[4.4.4 Сценарии синхронизации 20](#_Toc40688959)

[4.4.5 Контроль расхождения значений передаваемых данных 24](#_Toc40688960)

[4.4.6 Формирование синхронизированного значения параметра 27](#_Toc40688961)

[4.4.7 Контроль временных характеристик выполнения процесса синхронизации 30](#_Toc40688962)

[4.4.8 Обеспечение формирования синхронизированного значения 32](#_Toc40688963)

[4.4.9 Порядок использования МКО 33](#_Toc40688964)

[4.5 Статическая модель 36](#_Toc40688965)

[4.5.1 Статическая модель ПО 36](#_Toc40688966)

[4.5.2 Статическая модель компонентов 38](#_Toc40688967)

[4.6 Динамическая модель 42](#_Toc40688968)

[4.6.1 Взаимодействие потоков 42](#_Toc40688969)

[4.6.2 Взаимодействие режимов 43](#_Toc40688970)

[4.6.3 Взаимодействие компонентов 47](#_Toc40688971)

[4.7 Интерфейсы взаимодействия 52](#_Toc40688972)

[4.7.1 Компоненты 52](#_Toc40688973)

[4.7.2 Системные каталоги 79](#_Toc40688974)

[5 Обеспечение безопасности в ПО 82](#_Toc40688975)

[6 Исключения 91](#_Toc40688976)

[6.1 Обработка исключений 91](#_Toc40688977)

[6.1.1 Обработка отладочных исключений 93](#_Toc40688978)

[6.1.2 Отключение обработки исключений 93](#_Toc40688979)

[6.1.3 Отладочный режим функции sysAssertException 94](#_Toc40688980)

1. Введение
   1. Назначение документа

Этот документ описывает архитектуру разрабатываемого ПО и принципы ее построения. Документ предоставляет информацию для разработки требований к реализации ПО «Объектный контроллер привода стрелки. ОКПС-Е-К. ЕИУС.хххххх.ххх 01 92 01 01».

Документ распространяется на объектный контроллер привода стрелки ОКПС-Е-К (далее по тексту – прибор) для обеспечения безопасного управления стрелочным приводом путем формирования трехфазного напряжения для питания электродвигателей переменного тока в 9-ти проводных схемах управления.

* 1. Термины, определения и сокращения

Для применения данного документа имеют значение термины, определения и сокращения «Объектный контроллер привода стрелки. ОКПС-Е-К. ЕИУС.хххххх.ххх 96.01».

* 1. Рекомендации к ознакомлению

Этот документ:

* определяет архитектуру ПО прибора на уровне компонентов;
* не предназначен для разработки поддерживающего ПО (ПО стендов регулировки, ПО стендов проверки, ПО средств отладки);
* не предназначен для рассмотрения коммерческих вопросов;

Перед чтением данного документа следует ознакомиться с функциональными требованиями к ПО «Объектный контроллер привода стрелки ОКПС-Е-К. ЕИУС.хххххх.ххх 01 90 01 01».

Программные интерфейсы компонентов описаны в разделе 4.7 («Интерфейсы взаимодействия») данного документа.

* 1. Ссылки

1. Техническое задание на разработку «Аппаратуры управления и контроля стрелок»;
2. «Объектный контроллер привода стрелки. ОКПС-Е-К. Функциональные требования к ПО ЕИУС.хххххх.ххх 01 90 01 01»;
3. Unified Modeling Language Specification Version 2.5.
4. Аппаратные средства

Аппаратные средства описаны в разделе 2 «Объектный контроллер привода стрелки. ОКПС-Е-К. Функциональные требования к ПО ЕИУС.хххххх.ххх 01 90 01 01».

1. Внешние интерфейсы

Внешние интерфейсы описаны в разделе 3 «Объектный контроллер привода стрелки. ОКПС-Е-К. Функциональные требования к ПО ЕИУС.хххххх.ххх 01 90 01 01».



1. Логическая структура ПО
   1. Построение архитектуры
      1. Принципы обеспечения безопасности прибора
         1. Дублированная система с умеренными связями

Прибор построен как дублированная система с умеренными связями, которая включает в себя два одинаковых вычислительных канала (Master и Slave) на базе МК с одинаковыми программами. Работа обоих каналов синхронизирована как по времени выполнения, так и по данным. Синхронизация служит для одновременного считывания входных воздействий и выдачи управляющих воздействий. Для контроля работоспособности прибора каналы снабжены средствами самодиагностики.

Между Master и Slave реализованы два канала синхронизации: канал синхронизации времени выполнения и канал синхронизации данных. Синхронизация времени выполнения двух МК обеспечивается за счет приостановки потоков управления в более «быстром» МК и запуске их, когда более «медленный» МК достигнет заданной точки временно́й синхронизации. В точке временной синхронизации потоки управления максимально синхронизированы. По мере выполнения программы вычислители могут расходиться во времени до следующей точки временно́й синхронизации.

Синхронизация данных между МК обеспечивается при помощи обмена данными между ними и обработки этих данных так, чтобы в обоих МК использовалось одинаковое значение для дальнейшей обработки. Также канал синхронизации данных используется средствами самодиагностики для выполнения перекрестного контроля входных данных и основных параметров функционирования.

* + - 1. Обработка ошибок

В процессе работы программы могут возникать критические ошибки такие, как неисправность аппаратных средств, повреждения данных, неисправность МК, сбои в работе алгоритма программы, которые могут привести к опасным отказам прибора.

Обнаружение ошибок, которые могут привести к опасному отказу, выполняется системой самодиагностики в программных компонентах.

Обработка таких ошибок выполняется при помощи единого механизма утверждений (ASSERT), приводящего к переходу прибора в ЗС с сохранением информации о возникшей ошибке в ЭНП. Автоматический возврат прибора из ЗС в работоспособное состояние невозможен.

* + - 1. Безопасное состояние прибора

Прибор переходит в безопасное состояние при возникновении внешних негативных условий, при которых он не может гарантированно обеспечить формирование выходных сигналов управления двигателем стрелочного привода. Автоматический возврат прибора в работоспособное состояние возможен только при отсутствии внешних негативных условий.

* + 1. Принципы построения архитектуры

При построении архитектуры используются следующие положения:

1. отказ от использования операционных систем;
2. использование потоков управления;
3. инкапсуляция;
4. декомпозиция;
5. модульность;
6. иерархичность;
7. параллелизм;
8. централизация функций межканальной синхронизации.

Для построения ПО выбраны следующие базовые концепции:

1. квазипараллельные взаимодействующие процессы, выполняемые:
2. при обработке прерываний;
3. в компонентах основного цикла.
4. уровневая архитектура:
5. управление потоками;
6. управление режимами;
7. управление функциями прибора.
8. характеристики проектирования компонент с низкоуровневым интерфейсом:
9. внешний интерфейс компонента определен в одном модуле;
10. все остальные модули скрыты.

Используется шаблон проектирования «Фасад» (Facade).

* 1. Потоки управления

В программе реализовано четыре потока управления:

1. ГП;
2. ПВС;
3. ПСД;
4. поток обработки неиспользуемых прерываний.

Диаграмма последовательности потоков управления представлена на рисунке 1.

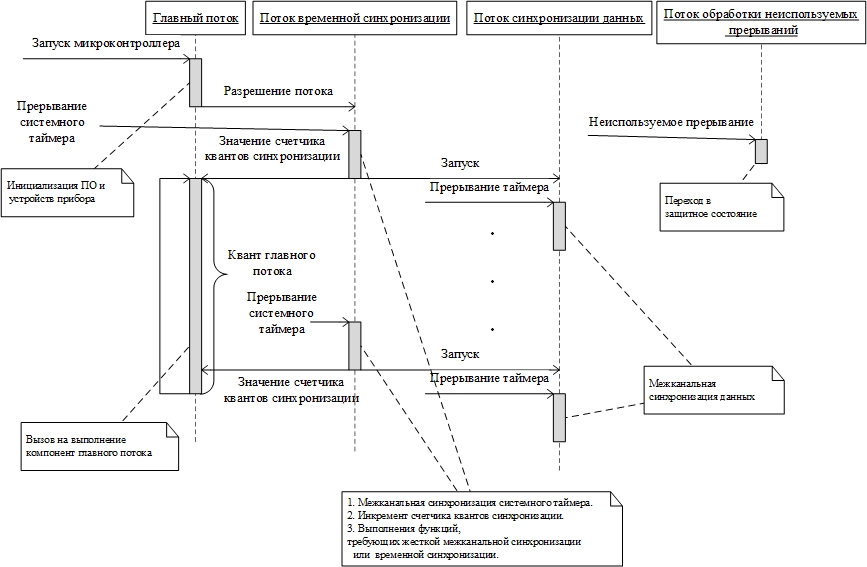


Рисунок 1 – Диаграмма последовательности потоков управления

* + 1. Главный поток

ГП получает управление после сброса МК, инициализирует компоненты ПО и периферийные устройства, запускает поток временной синхронизации и выполняет процессы основного цикла *main* с определенным периодом (квантом главного потока).

Квант ГП определяется конкретным количеством квантов синхронизации. Если все функции ГП выполняются до достижения заданного количества квантов синхронизации, ГП приостанавливается. Если ГП проработал дольше определенного периода, прибор переводится в ЗС. Выполнение компонентов основного потока является псевдопараллельным – компоненты последовательно вызываются в кванте ГП.

Для главного потока используется уровневая организация архитектуры. Структура представлена на рисунке 2.

Один и тот же компонент может использоваться в нескольких режимах.

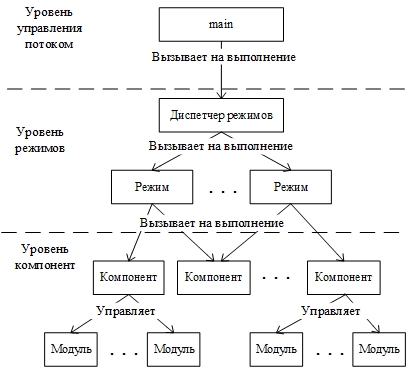


Рисунок 2 – Структура главного потока

Компонент предназначен для реализации функции прибора (ВИ) или для реализации сервиса, используемого другими компонентами (ПВИ). Компонент состоит из одного и более модулей. Модуль не может использоваться для нескольких компонентов. Модуль, имеющий тоже имя, что и компонент, содержит интерфейс для взаимодействия с другими компонентами.

В задачи компонента main входит:

1. управление потоками;
2. синхронизация ГП с ПВС;
3. вызов на выполнение диспетчера режимов;
4. контроль, что время работы кванта ГП не выходит за пределы допустимого.

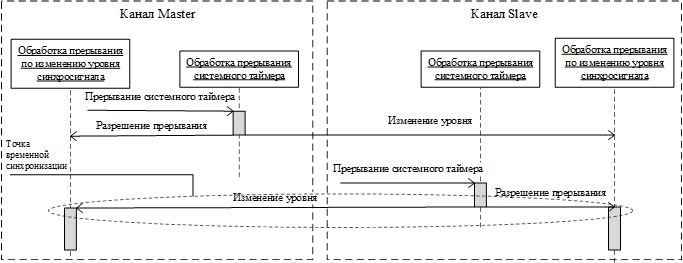
Диспетчер режимов представляет собой автомат состояний для вызова функций текущего режима. Режим является состоянием автомата и предназначен для вызова функций режима.

* + 1. Поток временно́й синхронизации

ПВС получает управление при достижении обоими МК точки синхронизации. Формирование точки синхронизации обеспечивается при помощи системы прерываний от таймера и по изменению уровня на входе МК. В данном потоке работают счетчик квантов синхронизации и функции, требующие высокой частоты вызова или высокой точности временно́й синхронизации. ПВС приостанавливает ГП на время своего выполнения. Поэтому функции, работающие в этом потоке, должны выполняться минимально возможное время, чтобы давать возможность работать остальным потокам. Данные должны подготавливаться в ГП, а в этом потоке должны приниматься синхронные решения и формироваться управляющие воздействия.

Поскольку место, в котором ГП прерывается ПВС, не определено, работа с общими данными в этих двух потоках разрешена только для данных, операции с которыми являются атомарными, или при помощи ПСД. Блокировка работы ПВС запрещена.

Принцип синхронизации по времени выполнения показан на рисунке 3.

Рисунок 3 – Принцип синхронизации по времени выполнения

Обработка прерывания по изменению уровня выполняется по двум условиям:

* это прерывание разрешено;
* есть изменение уровня синхросигнала.

Период синхронизации, в дальнейшем – квант синхронизации, отсчитывается системным таймером. На каждом кванте синхронизации меняются уровни на линиях синхронизации. Имеется две фазы синхронизации: по фронту сигнала и по спаду. При обработке прерывания по изменению уровня контролируется фаза синхронизации, если она не совпадает с ожидаемой, то происходит переход программы в ЗС. Также при обработке прерывания по изменению уровня в регистр системного таймера записывается константное значение равное времени кванта. Таким образом, проводится межканальная синхронизация системного таймера.

* + 1. Поток синхронизации данных

Для обмена данными, операции с которыми не являются атомарными, между ГП и ПВС предусмотрен ПСД. Этот поток получает управление после того, как ПВС отработал свой квант и приостановился до достижения следующей точки синхронизации. Механизм передачи управления реализован при помощи таймера МК. ПВС инициализирует и запускает таймер, а по прерыванию от этого таймера происходит передача управления. Данный поток всегда получает управление по завершению ПВС, поэтому одновременный доступ к общим данным этих потоков невозможен. Блокировка работы данного потока из ГП разрешена при помощи запрета прерываний таймера, передающего управление этому потоку. Обмен данными между ГП и ПВС должен происходить по следующему сценарию:

1. ГП блокирует работу ПСД, модифицирует общие данные и разблокирует его работу;
2. ПСД, получив управление, копирует измененные данные в буфер, доступный ПВС, и завершает свою работу;
3. ПВС, получив управление, обрабатывает данные из буфера.

Работа с данными может вестись в обоих направлениях. Для обмена данными между ПВС и ГП должен использоваться следующий сценарий:

1. ПВС помещает данные в буфер ПСД;
2. ПСД копирует из буфера данные в область памяти, которая используется в ГП;
3. ГП блокирует работу ПСД, читает общие данные и разблокирует его.

ПСД не должен выполнять никаких функций, кроме буферизации данных.

Этот механизм работает для синхронизации данных ГП. Для синхронизации данных, используемых в ПВС, должен использоваться следующий сценарий:

1. провести синхронизацию локальной переменной (см. выше), результат сохранить во временной переменной;
2. дать указание ПВС обновить синхронизированную переменную;
3. ПВС записывает данные из временной переменной в синхронизированную переменную.
   * 1. Поток обработки неиспользуемых прерываний

Поток обработки неиспользуемых прерываний предназначен для обработки немаскируемых прерываний, которые могут возникать в случае обнаружение сбоев в работе МК. По каждому вектору прерывания, который имеется в МК, определяется обработчик, который переводит прибор в ЗС.

* 1. Обмен данными между потоками

В системе предполагается четыре потока управления:

* ГП;
* ПВС;
* ПСД;
* поток обработки неиспользуемых прерываний.

Поток обработки неиспользуемых прерываний предназначен для выявления сбоев МК и в работе не используется, поэтому организация взаимодействия других потоков с этим потоком не рассматривается.

Основная работа ведется в потоке временной синхронизации и в главном потоке. Поскольку блокировка ПВС запрещена, обмен между ГП и ПВС возможен только данными, работа с которыми ведется атомарными операциями. Работа с более сложными структурами данных между этими потоками является небезопасной. Для безопасного доступа к данным между ГП и ПВС введен отдельный поток-«посредник» ПСД. Для безопасного обмена данными используются следующие два механизма управлениями потоками:

* синхронная работа ПВС и ПСД;
* блокировка ГП получения управления ПСД.

Синхронная работа ПВС и ПСД достигается за счет последовательной передачи управления потокам. ПВС получает управление на каждом кванте синхронизации и после завершения работы запускает таймер, который передает управление ПСД. ПСД, выполнив свои функции, останавливается пока ПВС снова не передаст ему управление. Таким образом, потоки выполняются строго друг за другом и никогда не пересекаются во времени.

Блокировка ГП получения управления ПСД осуществляется при помощи запрета прерывания от таймера при помощи которого ПСД получает управление. Снятие блокировки осуществляется при помощи разрешения запрещенного прерывания. Когда ГП обращается к общим данным, он блокирует ПСД, и тот в это время не может получить управление, следовательно, не может обратиться к общим данным. После завершения работы с общими данными ГП снимает блокировку и ПСД получает доступ к данным.

Циклограмма работы потоков без блокировки приведена на рисунке 4. Циклограмма работы потоков во время блокировки приведена на рисунке 5.



Рисунок 4 – Циклограмма работы потоков без блокировки



Рисунок – Циклограмма работы потоков во время блокировки

Из рисунка видно, что ПСД в двух МК может получать управление асинхронно, в зависимости от того, в какой момент времени и насколько ГП заблокировал его работу. Однако, он должен получить управление и завершить свою работу до того как ПВС получит управление на следующем кванте временной синхронизации.

* 1. Синхронизация данных между процессорами
     1. Синхронизация данных

Синхронизация данных – это процесс ликвидации различий между двумя копиями данных в двух МК, выполняющих одинаковую программу. Предполагается, что два МК, выполняющие одинаковую программу, могут считывать данные от внешних систем, которые могут изменять свои значения независимо друг от друга. Кроме того, различия значений входных данных в двух МК могут быть обусловлены тем, что считывание этих значений может происходить в различный момент времени из-за разницы в тактовой частоте, а также из-за возможности выполнения отдельных функций, специфических только для одного из МК.

Изменения значений внешних данных рассматриваются как события, на которые должно отреагировать ПО. Из-за того, что изменения значений данных разные МК могут зафиксировать в разные моменты времени, они отреагируют на эти события асинхронно. Для принятия одинаковых решений обоими МК, а также для синхронизации реакции на внешние события, необходимо синхронизировать данные и использовать в дальнейшей обработке результат синхронизации.

Для синхронизации событий в обоих МК к процедуре синхронизации данных выдвигается дополнительное требование – обновление синхронизируемых данных в обоих МК должно происходить одновременно в начале нового цикла ГП. Тогда реакция пользовательского ПО на события будет происходить с точностью до одного цикла ГП.

Синхронизация данных между МК должна осуществляется при помощи компонента «Модуль межканального обмена» (МКО) – InterChannel.

* + 1. Синхронизируемые параметры

Данные, которые синхронизируются при помощи МКО, называются параметрами. При помощи МКО могут синхронизироваться данные только тех типов, которые могут быть приведены к типу param\_t, который определен в интерфейсном модуле компонента InterChannel, как беззнаковое целое разрядностью 16 бит.

* + 1. Идентификатор синхронизируемого параметра

Каждому параметру должен соответствовать свой уникальный идентификатор, который определен в интерфейсном системном модуле InterChannelId как часть перечислимого типа *InterChannelId* и должен быть расположен между значениями *eInterChannelIdBegin* и *eInterChannelIdCount*. Значение *eInterChannelIdCount* определяет общее количество синхронизируемых параметров.

Примечание

Не допускается существование двух идентификаторов с одинаковыми значениями.

* + 1. Сценарии синхронизации

Синхронизация параметров может осуществляться по нескольким сценариям (алгоритмам). Сценарий синхронизации – это группа последовательных операций над значениями параметров, сформированными в разных МК и имеющими общий идентификатор. Группа представляет собой логическую единицу работы с данными. Сценарий синхронизации может быть выполнен либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других сценариев для параметров с другими идентификаторами, либо не выполнен вообще и тогда он не должен произвести никакого эффекта.

Доступные сценарии синхронизации должны быть определены в интерфейсном модуле компонента InterChannel как перечисляемый тип InterChannelScript, который должен содержать следующие значения:

* *eScriptNil* − сценарий не определен;
* *eScriptSync* − синхронизация параметра;
* *eScriptTransmit* − передача параметра;
* *eScriptLocal* – опубликовать параметр без отправки в соседний канал;
* *eScriptDebug* − передача параметра без контроля и синхронизации;
* *eScriptChVal* – процедура синхронизации по изменению значения;
* *eScriptCount* − количество сценариев синхронизации.
  + - 1. Сценарий синхронизации «Сценарий не определен»

Данный сценарий должен быть задан при создании компонента МКО. При попытке синхронизации параметра с таким сценарием прибор должен быть переведен в ЗС.

* + - 1. Сценарий синхронизации «Синхронизация параметра»

Сценарий синхронизации «Синхронизация параметра» позволяет синхронизировать события и данные в двух МК. Данный сценарий считается завершенным только тогда, когда выполнены следующие операции:

* оба МК обратились к МКО с запросом на синхронизацию параметра;
* оба МК обменялись значениями параметров и получили подтверждение о приеме данных соседом;
* выполнен контроль расхождения значений передаваемых данных;
* сформировано синхронизированное значение параметра;
* установлен признак завершения синхронизации.

Сценарий требует того, что на каждый запрос синхронизации в одном МК должен сформироваться запрос в другом МК. Допускается расхождение формирования запросов по времени, которое должно контролироваться. Не допускается формирование двух и более событий по одному параметру в одном МК, пока на первое не возникло соответствующее событие в другом МК.

Если при контроле расхождений было установлено что расхождение превышает допустимое, сценарий считается завершенным, но синхронизация не выполненной (синхронизированное значение не обновляется и признак завершения синхронизации не формируется).

Если событие возникло только в одном МК или в одном МК возникло большее число событий, чем в другом, прибор должен быть переведен в ЗС по причине того, что сценарий не может быть завершен. Время перевода прибора в ЗС определяется пользовательским ПО, учет его ведется с момента возникновения нового события в одном из МК.

Если не было получено подтверждение о приеме данных, данный сценарий должен перевести прибор в ЗС, выполнив определенное количество попыток передать данные.

* + - 1. Сценарий синхронизации «Передача параметра»

Сценарий синхронизации «Передача параметра» позволяет синхронизировать события и данные в двух МК, если источник этих данных подключен только к одному МК.

Данный сценарий считается завершенным только тогда, когда выполнены следующие операции:

* один из МК обратился к МКО с запросом на синхронизацию параметра;
* значение параметра было передано в соседний МК и получено подтверждение о приеме;
* сформировано синхронизированное значение параметра;
* установлен признак завершения синхронизации.

Сценарий обеспечивает одновременное формирование синхронизированного значения параметра в обоих МК. В МК-источнике синхронизированное значение должно быть сформировано только тогда, когда МК-получатель подтвердил прием данных, а в МК-получателе – только после того как передал подтверждение приема данных. Обновленное значение в обоих МК должно быть сформировано на одном цикле ГП.

Если не было получено подтверждения о приеме данных, данный сценарий должен перевести прибор в ЗС, выполнив определенное количество попыток передать данные.

* + - 1. Сценарий синхронизации «Опубликовать параметр без отправки в соседний канал» (Режим заглушки)

Сценарий синхронизации «Опубликовать параметр без отправки в соседний канал» формирует синхронизированное значение сразу после получения запроса, без отправки параметра соседнему МК. В качестве синхронизированного значения используется значение, которое было в запросе на синхронизацию. Данный сценарий может быть использован при отладке ПО в однопроцессорном режиме.

* + - 1. Сценарий синхронизации «Передача параметра без контроля и синхронизации» (Отладочная передача параметра)

Сценарий синхронизации «Передача параметра без контроля и синхронизации» формирует синхронизированное значение после того, как параметр был передан соседнему каналу и получено подтверждение. В качестве синхронизированного значения используется значение, которое было в запросе на синхронизацию. При приеме соседний канал просто игнорирует принятые значения. Данный сценарий может быть использован при отладке ПО в двухпроцессорном режиме для вывода данной информации на ПЭВМ, подключенной к шине, по которой выполняется передача параметров между МК.

* + - 1. Сценарий синхронизации «Синхронизация по изменению значения»

Сценарий синхронизации «Синхронизация по изменению значения» предназначен для синхронизации непериодических сигналов. Данный сценарий не выполняет синхронизацию событий, а только ликвидирует различия между значениями параметра в двух МК. После поступления запроса на синхронизацию от одного из МК, осуществляется передача нового значения соседнему МК. После подтверждения приема соседним МК выполняется контроль расхождения значений, переданных в соседний канал и ранее присланных соседним каналом. В свою очередь, соседний МК, получив данные, передает подтверждение приема и также выполняет контроль расхождения значений, принятых от соседнего МК и ранее переданных в соседний МК.

Если значение не превышает допустимые пределы, на основе проверенных значений формируются синхронизированное значение и признак окончания синхронизации – сценарий считается завершенным. Если при контроле расхождений было установлено, что расхождение превышает допустимое, сценарий считается завершенным, но синхронизация не выполненной. Синхронизированное значение не обновляется и признак завершения синхронизации не формируется.

Таким образом, оба МК оперируют одинаковыми значениями независимо от количества событий, зафиксированных одним или другим МК.

* + 1. Контроль расхождения значений передаваемых данных

Значения параметра, зафиксированные в разных МК, могут отличаться друг от друга. Прежде чем сформировать синхронизированное значение необходимо оценить насколько эти значения отличаются друг от друга и допустимо ли на их основании формировать его. За оценку значений параметра, зафиксированных в разных МК, отвечает процедура контроля. Процедура контроля – это набор критериев, в соответствии с которыми проводится оценка значений параметров, в результате которой принимается решение о допустимости формирования синхронизированного значения. Поскольку параметры могут носить разный физический смысл, в МКО должно быть предусмотрено несколько разных процедур контроля, которые для каждого параметра индивидуально выбираются пользовательским ПО.

Доступные процедуры контроля синхронизации должны быть определены в интерфейсном модуле компонента InterChannel.h как перечисляемый тип *InterChannelProcCheck*, который должен содержать следующие значения:

* *eProcCheckNil* − не выполняется никаких действий с параметром;
* *eProcCheckOff* − нет контроля (применяется при регулировке аппаратуры или передаче параметра);
* *eProcCheckEqual* − абсолютное совпадение данных;
* *eProcCheckMask* − совпадение данных по маске;
* *eProcCheckDeltaX* − абсолютная разность находится в пределах «deltaХ» (значение);
* *eProcCheckDeltaP* − абсолютная разность находится в пределах «deltaP» (процент).
  + - 1. Процедура контроля «Не выполняется никаких действий с параметром»

Данная процедура контроля должна быть задана при создании компонента МКО. При попытке оценки значений параметра с такой процедурой контроля прибор должен быть переведен в ЗС.

* + - 1. Процедура контроля «Нет контроля»

Данная процедура контроля применяется при регулировке аппаратуры или передаче параметра. При попытке оценки значений параметра с такой процедурой контроля результат такой оценки всегда должен быть положительный.

* + - 1. Процедура контроля «Абсолютное совпадение данных»

При попытке оценки значений параметра с такой процедурой контроля результат будет положительным, только если значения параметра, зафиксированные в разных МК, равны друг другу.

* + - 1. Процедура контроля «Совпадение данных по маске»

При попытке оценки значений параметра с такой процедурой контроля результат будет положительным, только если в значениях параметра, зафиксированных в разных МК, совпадают биты, заданные в маске. Маска при помощи логической операции «И» накладывается на оба значения, после чего сравниваются результаты.

* + - 1. Процедура контроля «Абсолютная разность находится в пределах «deltaХ» (значение)»

При попытке оценки значений параметра с такой процедурой контроля результат будет положительным, только если разница между значениями параметра, зафиксированными в разных МК не превышает заданную.

* + - 1. Процедура контроля «Абсолютная разность находится в пределах «deltaP» (процент)»

При попытке оценки значений параметра с такой процедурой контроля результат будет положительным, только если разница между значениями параметра, зафиксированными в разных МК, не будет превышать заданную в процентном соотношении.

При оценке разницы значений в процентном соотношении должен применяться подход, при помощи которого производится оценка относительной погрешности. Относительная погрешность измерения – отношение абсолютной погрешности измерения к опорному значению измеряемой величины, в качестве которого может выступать, в частности, её истинное или действительное значение.

В качестве опорного значения принимается одно из действительных значений параметра, зафиксированных в разных МК. Однако, в некоторых случаях, когда действительное значение лежит в начале диапазона измерения, отклонение в 10% может оказаться очень маленькой величиной и тогда любая помеха или шум могут быть восприняты как отказ. Поэтому предусмотрена возможность указать пользовательскому ПО опорное значение, которое будет использоваться, если значения синхронизируемого параметра окажутся ниже этого значения. Если опорное значение задано нулём, в качестве опорного всегда будет использоваться действительное значение синхронизируемого параметра.

* + 1. Формирование синхронизированного значения параметра

После успешного прохождения процедуры контроля на основании двух значений, полученных от разных МК, должно быть сформировано одно значение, которое в обоих каналах будет одинаковым – это значение называется синхронизированным. Процедуру формирования синхронизированного значения далее будем называть процедурой синхронизации. Поскольку параметры могут носить разный физический смысл, в МКО должно быть предусмотрено несколько разных процедур синхронизации, которые для каждого параметра индивидуально выбираются пользовательским ПО.

После выполнения процедуры синхронизации, МКО сформирует признак выполнения синхронизации, который будет сброшен при поступлении нового запроса для этого параметра.

Доступные процедуры контроля синхронизации должны быть определены в интерфейсном модуле компонента InterChannel.h как перечисляемый тип *InterChannelProcSync*, который должен содержать следующие значения:

* *eProcSyncNil* − не выполняется никаких действий с параметром;
* *eProcSyncOff* − нет синхронизации (в качестве синхронизируемых данных возвращается значение своего канала);
* *eProcSyncHi* − выбор большего значения;
* *eProcSyncLo* − выбор меньшего значения;
* *eProcSyncAverage* − выбор арифметически среднего значения;
* *eProcSyncAND* − выбор по «AND»;
* *eProcSyncOR* − выбор по «OR»;
* *eProcSyncEqual* – обновление значений параметров, если они совпадают.
  + - 1. Процедура синхронизации «Не выполняется никаких действий с параметром»

Данная процедура синхронизации должна быть задана при создании компонента МКО. При попытке формирования синхронизированного значения параметра с такой процедурой, прибор должен быть переведен в ЗС.

* + - 1. Процедура синхронизации «Нет синхронизации»

При выполнении этой процедуры всегда в качестве синхронизированного значения используется значение, зафиксированное текущим МК. Данная процедура может использоваться при использовании сценариев синхронизации таких как «Передача параметра без контроля и синхронизации», «Опубликовать параметр без отправки в соседний канал» (Режим заглушки) и т.д.

* + - 1. Процедура синхронизации «Выбор большего значения»

При выполнении этой процедуры всегда в качестве синхронизированного значения используется большее значение.

* + - 1. Процедура синхронизации «Выбор меньшего значения»

При выполнении этой процедуры всегда в качестве синхронизированного значения используется меньшее значение.

* + - 1. Процедура синхронизации «Выбор арифметически среднего значения»

При выполнении этой процедуры всегда в качестве синхронизированного значения используется среднее арифметическое значений, зафиксированных обоими МК.

* + - 1. Процедура синхронизации «Выбор по «AND»

При выполнении этой процедуры всегда в качестве синхронизированного значения используется значение, полученное после выполнения логической операции конъюнкции над значениями, зафиксированными обоими МК.

* + - 1. Процедура синхронизации «Выбор по «OR»

При выполнении этой процедуры всегда в качестве синхронизированного значения используется значение, полученное после выполнения логической операции дизъюнкции над значениями, зафиксированными обоими МК.

* + - 1. Процедура синхронизации «Обновление значений параметров, если они совпадают»

При выполнении этой процедуры всегда в качестве синхронизированного значения может быть выбрано любое из значений параметра, поскольку данная процедура будет выполнена только при выполнении процедуры контроля «Абсолютное совпадение данных».

Примечание: при выборе пользовательским ПО данной процедуры синхронизации МКО вместе с ней должна быть указана процедура контроля «Абсолютное совпадение данных». Использование данной процедуры синхронизации с другими процедурами контроля запрещено и должно привести к переходу прибора в ЗС.

* + 1. Контроль временных характеристик выполнения процесса синхронизации

В процессе синхронизации параметра должен выполняться контроль этапов его выполнения по времени. За выполнение контроля должен отвечать МКО. Пользовательское ПО должно иметь возможность настраивать этот контроль для каждого синхронизируемого параметра.

Модуль МКО в процессе синхронизации должен выполнять контроль следующих временных параметров:

* время получения параметра от обоих каналов (Т1), мс;
* время формирования синхронизированного параметра (Т2), мс;
* период поступления запросов на синхронизацию (Т3), мс.

Контроль каждого из времен может быть выключен, если установить значение контролируемого времени равным нулю.

Единица контроля времени соответствует периоду выполнения цикла ГП.

* + - 1. Время получения параметра от обоих каналов (Т1)

Время получения параметра от обоих каналов предназначено для контроля выполнения сценария синхронизации «Синхронизация параметра» для контроля расхождения во времени событий, сформировавших запросы синхронизации параметра в разных МК. Отсчет времени начинается с того момента как один из МК обратился с запросом синхронизации параметра к МКО. Останавливается отсчет времени при поступлении запроса на синхронизацию того же параметра от второго МК, что является критерием для завершения сценария. Если время достигнет значения заданного пользовательским ПО, прибор должен быть переведен в ЗС.

* + - 1. Время формирования синхронизированного параметра (Т2)

Время формирования синхронизированного параметра определяет время, на протяжении которого оба МК могут фиксировать значения параметра, которые не проходят процедуру контроля и МКО не формирует синхронизированное значение. Отсчет времени начинается с того момента, как процедура контроля выдала отрицательный результат. Останавливается, когда процедура контроля выдала положительный результат. Если время достигнет значения заданного пользовательским ПО, прибор должен быть переведен в ЗС.

Время Т2 должно задаваться исходя из тех соображений, что для того, чтобы процедура контроля была вызвана повторно, необходимо, чтобы МК послали очередной запрос синхронизации того параметра, который не прошел эту процедуру в первый раз. То есть, если параметр синхронизируется с фиксированным периодом и, исходя из прикладной задачи, допускается рассогласование значений на протяжении двух периодов, то это время должно быть равным или более двух периодов.

* + - 1. Период поступления запросов на синхронизацию (Т3)

Для параметров, которые должны синхронизировать свои значения с заданным периодом, в МКО должна быть предусмотрена возможность контроля периода поступления запросов на синхронизацию значений параметров. Отсчет времени начинается с того момента как в МКО поступит запрос на синхронизацию параметров. При поступлении очередного запроса счетчик времени сбрасывается. Если время достигнет значения заданного пользовательским ПО, прибор должен быть переведен в ЗС.

* + 1. Обеспечение формирования синхронизированного значения

Синхронизированное значение должно формироваться одновременно в начале нового цикла ГП в обоих МК. Опрос периферии МК осуществляется в ПВС и ПСД.

Для обеспечения заданной точности МКО должно накапливать информацию, принятую от соседнего канала, на протяжении всего цикла ГП, а в начале следующего – обработать накопившуюся информацию (информацию о подтверждении приема и принятые значения параметров от соседнего канала) и завершить сценарии для тех параметров, имеющих все условия для этого. Если сценарии завершатся без ошибок, необходимо сформировать синхронизированное значение и признак завершения синхронизации.

Поскольку начало цикла ГП синхронизируется с циклами ПВС, оба МК начнут обрабатывать собранную информацию одновременно. Далее пользовательское ПО при обращении к МКО сможет определить, для какого параметра завершились сценарий и синхронизация после чего сможет использовать синхронизированное значение в дальнейшей обработке или, если сценарий завершился с ошибкой, принять меры. Благодаря такому механизму на каждом цикле ГП пользовательское ПО в обоих МК будет работать с одинаковыми данным.

Узким местом в данной схеме является момент обработки информации МКО в ГП. Время обработки может занимать времени больше, чем период работы ПВС и ПСД. Из-за этого в процессе обработки могут быть приняты новые данные от соседнего МК или подтверждение приема. Эти данные могут попасть в обработку асинхронно в одном канале уже на текущем цикле ГП, а в соседнем только на следующем. Для устранения данной проблемы в МКО должно быть введено ограничение на передачу данных перед началом нового цикла. Ограничение запрещает выдавать данные в CAN шину, если до начала нового цикла осталось времени меньше чем на передачу двух сообщений (одно сообщение с одной стороны, другое – с другой). Данное ограничение обеспечит отсутствие формирования событий приема данных и подтверждения приема, а, значит, в начале главного цикла в обоих каналах будут обработаны одинаковые события и данные.

Определять достаточно времени до завершения цикла ГП должно пользовательское или системное ПО и передавать в МКО запрет или разрешение на передачу данных.

* + 1. Порядок использования МКО
       1. Обеспечение функционирования МКО

Основная работа МКО осуществляется в ГП. Однако, работа с периферией МК при помощи которой осуществляется обмен между МК должна осуществляться в ПВС для синхронной фиксации состояний обмена данными и в ПСД – для безопасного обмена данными с ГП. Поскольку МКО должен работать в трех потоках управления, для каждого из них должны быть реализованы отдельные функции, которые будут вызываться на каждом цикле каждого потока обеспечивать функционирование модуля. Для работы в ГП должна быть реализована функция ***InterChannel\_run()***, для работы в потоке ПВД – ***InterChannel\_runDrv()***, для работы в потоке ПСД – ***InterChannel\_runCommunication()***. Пользовательское ПО в свою очередь должно обеспечить вызов в этих функций на каждом цикле соответствующего потока.

Для того чтобы обеспечить ограничение на передачу данных перед началом нового цикла, пользовательское ПО должно сообщить может ли МКО выдавать данные на шину или нет. Для этого при вызове функции ***InterChannel\_runCommunication()*** ей в качестве параметра необходимо передать разрешение передачи данных.

* + - 1. Подготовка к компиляции МКО

Перед сборкой проекта должны быть определены все идентификаторы синхронизируемых параметров проекта. Они должны быть добавлены в структуру *InterChannelId* в модуле InterChannelId (см. 4.4.3).

Взаимодействие с периферийными модулями МК при помощи которых обеспечивается обмен данными для синхронизации параметров, должен осуществляется при помощи драйвера ArrayIoDriver.

Для работы с ПСД (блокировки и разблокировки его работы из ГП при обращении к общим данным) должны быть использованы следующие макроопределения, определенные в модуле MainRegisters:

* CAN\_TIMER\_DISABLE\_INTERRUPT;
* CAN\_TIMER\_IS\_ENABLE\_INTERRUPT;
* CAN\_TIMER\_SET\_ENABLE\_INTERRUPT( en ).
  + - 1. Настройка МКО

Перед использованием модуля МКО, должен быть проинициализирован драйвер, используемый для обмена данными между МК. В данном проекте это EcanAck\_driverPIC33, а функция инициализации - ***EcanAck\_ctor()***. Функция инициализации этого драйвера должна вернуть указатель на структуру типа ArrayIoDriver.

После чего должны быть вызвана функция инициализации ***InterChannel\_ctor()***, которой в качестве параметра должна быть передана структура, которую вернул драйвер. После инициализации МКО должен быть готов работе.

* + - 1. Настройка параметров синхронизации

После инициализации модуля МКО всем параметрам в качестве сценария синхронизации задан сценарий «Сценарий не определен». При попытке синхронизации параметра с таким сценарием прибор должен быть переведен в ЗС. Поэтому, прежде чем отправлять запрос на синхронизацию, пользовательское ПО должно настроить параметры синхронизации синхронизируемого параметра.

Для настройки параметра в МКО должна быть предусмотрена функция ***InterChannel\_setParamSettings()***. Данная функция должна вызываться для каждого параметра перед отправкой первого запроса. Функции должны быть переданы следующие настройки:

* сценарий синхронизации, процедура контроля;
* процедура синхронизации;
* значение допустимого расхождения;
* опорное значение, которое будет использоваться при выполнении процедуры контроля «Абсолютная разность находится в пределах «deltaP» (процент)»;
* значения контролируемых временных параметров Т1, Т2, Т3.

Примечание

Если какой-то из параметров не планируется использовать, он должен быть передан равным нулю.

* + - 1. Запросы синхронизации

Любой сценарий синхронизации начинается с поступления запроса на синхронизацию. Запросы синхронизации формируются при помощи функции МКО ***InterChannel\_synchronize()***, которая принимает идентификатор параметра и его значение, зафиксированное МК.

При поступлении запроса синхронизации сбрасывается признак завершения синхронизации *InterChannel\_isSynchronized* и устанавливается признак процесса выполнения синхронизации параметра (далее по тексту – признак выполнения сценария синхронизации) *InterChannel\_isHandling*.

* + - 1. Проверка завершения сценария

Сценарий синхронизации может быть завершен как успешно (выполнен контроль расхождений, сформированы синхронизированное значение и признак завершения синхронизации), так и с ошибкой (контроль расхождений сообщил, что нельзя формировать синхронизированное значение на основании синхронизируемых значений). Поэтому для того, чтобы определить что сценарий завершился, в МКО должен быть предусмотрен признак выполнения сценария синхронизации ***InterChannel\_isHandling***, который говорит о том, что сценарий в процессе выполнения.

* + - 1. Проверка завершения синхронизации

Синхронизация считается завершенной тогда, когда в обоих МК сформировано синхронизированное значение. Если сценарий завершен, но значение не сформировано, синхронизация считается невыполненной. Для того чтобы определить выполнена ли синхронизация, в МКО должен быть предусмотрен признак завершения синхронизации ***InterChannel\_isSynchronized***.

* + - 1. Результат синхронизации

В случае успешного выполнения сценария синхронизации формируется синхронизированное значение, которое может быть прочитано функцией МКО ***InterChannel\_getData***. Если сценарий завершился с ошибкой, тогда синхронизированное значение не изменяется и функция ***InterChannel\_getData*** должны возвращать прежнее синхронизированное значение.

* 1. Статическая модель
     1. Статическая модель ПО

Статическая модель ПО представлена на рисунке 6.

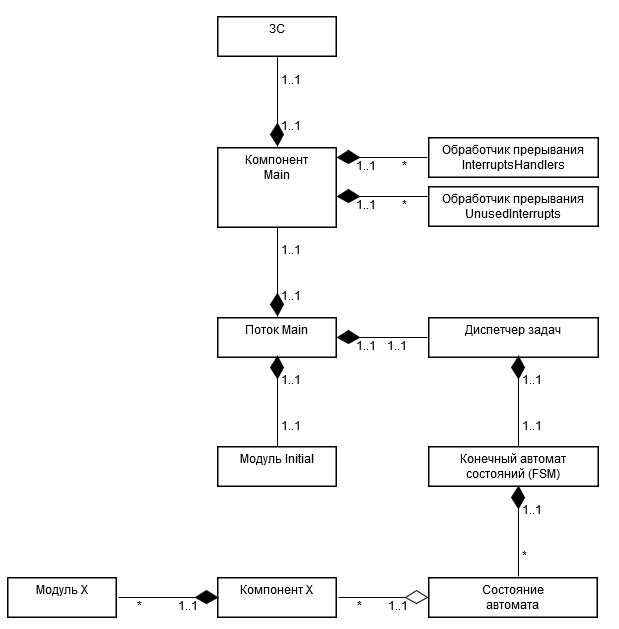


Рисунок – Статическая модель ПО

Компонент *Main* предназначен для управления компонентами потока.

Модуль *InterruptsHandlers* предназначен для обработки прерываний. Модуль *UnusedInterrupts* предназначен для обработки неиспользуемых прерываний. Состоят из набора функций обработки прерываний, имена функций обработки определяются типом МК и используемым компилятором.

Модуль *ModeProtection* реализует режим ЗС.

Модуль *Main* реализует работу ГП.

Модуль *Initial* обеспечивает первоначальную инициализацию периферии, переменных и всех модулей компонентов. Для первичной временной синхронизации каналов используется модуль *TimeBeginSynchronization*.

*Диспетчер задач* управляет режимами работы прибора (в соответствии с рисунком 9).

Модуль *Fsm* является конечным автоматом состояний.

* + 1. Статическая модель компонентов

Статическая модель объектов представлена на рисунке 8. Эта модель действенна как для канала Master, так и для канала Slave. Работа компонентов проводится в главном потоке. Работа отдельных модулей и функций компонент может проходить в других потоках.

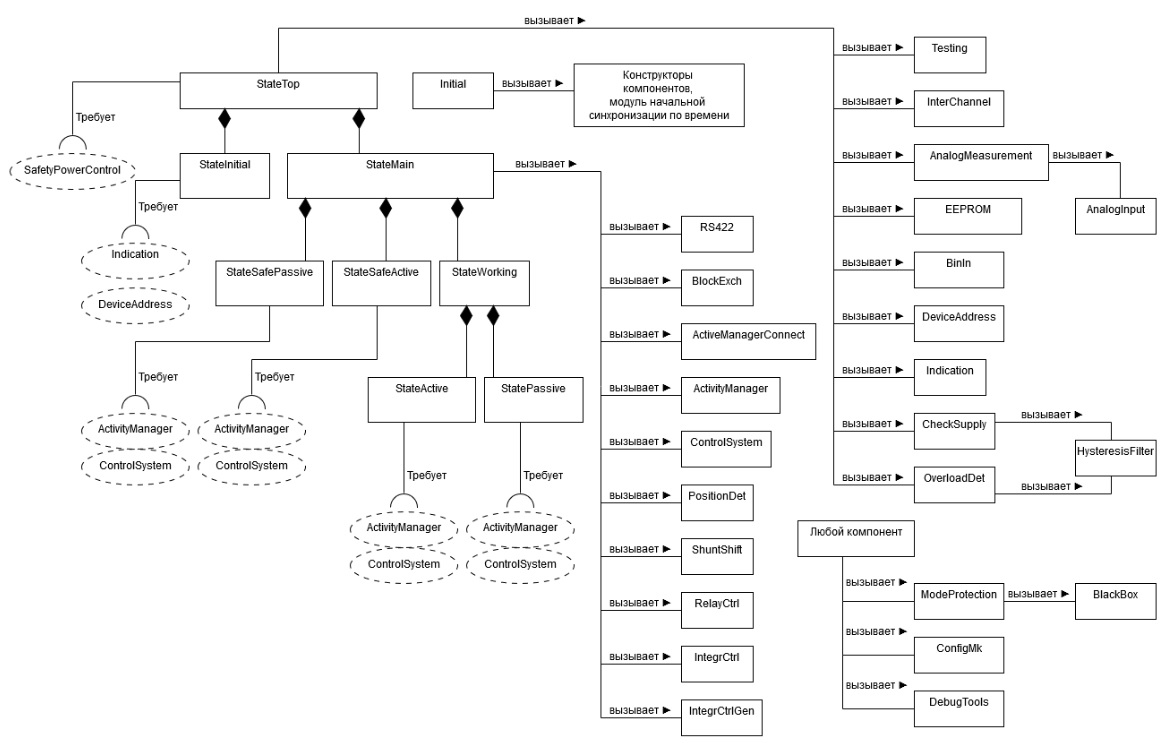


Рисунок – Статическая модель компонентов

Для прибора предусматриваются следующие компоненты и режимы в соответствии с рисунками 7 и 8:

* *StateMain* – управление главным потоком, ПВС и ПСД;
* *FSM* – конечный автомат состояний;
* *Initial* – компонент «Инициализация периферии микроконтроллера»;
* *StateSafePassive*– режим «Пассивное безопасное состояние»;
* *StateSafeActive*– режим «Активное безопасное состояние»;
* *StateActive* – режим «Активное рабочее состояние»;
* *StatePassive* – режим «Пассивное рабочее состояние»;
* *ModeProtection* – режим «Защитное состояние»;
* *StateInitial* – режим «Инициализация работы прибора»;

Режимы обеспечивают выполнение определенных наборов функций прибора, которые реализуются в следующих компонентах:

* *Sheduler* – диспетчер режимов работы;
* *RS422* – обмен по интерфейсу RS-422 данными с управляющей системой;
* *Main* (обработчики прерывания таймера и изменения сигнала по уровню) – межканальная синхронизация по времени;
* *InterChannel* – межканальная синхронизация по данным;
* *ActivityManager* (модуль *BlockExch) –* межприборный обмен «Основной - Резервный» (компонент);
* *BinIn* – чтение входных дискретных сигналов;
* *AnalogInput* **(**включающий модули *Adc*, *AnalogInput*, *DigitalFilter*) – работа со встроенным АЦП, обработка аналоговых сигналов и цифровая фильтрация;
* *AnalogMeasurement* – обработка аналоговых измерений данных, полученных из компонента *AnalogInput*;
* *ActivityManager* – управление активностью прибора;
* *Indication* (модуль *LedFailure*) – управление светодиодом «Отказ»;
* *RelayCtrl* – управление и контроль состояния реле, переключающего выходы (РПВ);
* *Testing* (модули *CheckCallFunctions, CheckCPU, FlashCheck, CheckRAM, CheckRegisters, ControlMK*) – проверка работоспособности МК, целостности и идентичности содержимого флэш-памяти обоих МК прибора;
* *Indication* – управление светодиодными индикаторами, расположенными на передней панели прибора;
* *Main* (модуль *ControlSystem*) **–** верхний уровень управляющей логики, включающий в себя реализацию выполнения приказов от УС и ответ статусами в УС, также сбор данных от основных компонентов прикладного уровня;
* *ConfigMK* **–** определение канала, в котором установлен МК (Master или Slave);
* *ModeProtection* **–** переход и снятие в ЗС;
* *BlackBox* – формирование и чтение информации «чёрного ящика»;
* *CheckSupply* – проверка состояния внешних источников питания 24 В и 220 В;
* *DebugTools* (модули*DebugTools* и *Tracing*) – средства отладки прибора и трассировки (записи в EEPROM) отладочной информации;
* *DeviceAddress –* определение адреса и конфигурации прибора и его конфигурации;
* *Eeprom* – работа с внешней энергонезависимой памятью EEPROM, содержащей данные чёрного ящика и трассировки;
* *HystFltr* – антидребезговый фильтр для цифровых сигналов;
* *Initial* и *Мain* (модули *CheckCauseReset*, *IdentHex*, *TimeBeginSynchronization*) – инициализация периферии МК, начальная синхронизация межпроцессорного обмена, чтение данных о версии ПО прибора, его типе;
* *IntegrCtrl* (модули *IntegrCtrl*, *IntegrCtrlGen*) – контроль рабочих цепей, т.е. цепей управления электродвигателем стрелочного привода;
* *SafetyPowerControl –* управление и контроль безопасного источника питания драйвера силовых ключей;
* *ShuntShift* **(**модули *ShuntShiftGen, ShuntShiftMotor, ShuntShiftNineWire*)  – управление генерацией трёхфазного сигнала, управление асинхронным трёхфазным электродвигателем, управление стрелочным переводом;
* *PositionDet* (модули *PosDetGenerator*, *PositionDetNineWire*)  –  генерация напряжения контрольных цепей и определение положения стрелки;
* *OverloadDet* – определение перегрузок по току в рабочих и в контрольных цепях.
  1. Динамическая модель

Вызов компонент на выполнение соответствует статической модели. Далее в этом разделе рассматривается информационное взаимодействие компонент.

* + 1. Взаимодействие потоков

Взаимодействие главного потока и потоков синхронизации показано на рисунке 7. Поток обработки неиспользуемых прерываний с другими потоками не взаимодействует, если не считать вызова компонента режима ЗС.

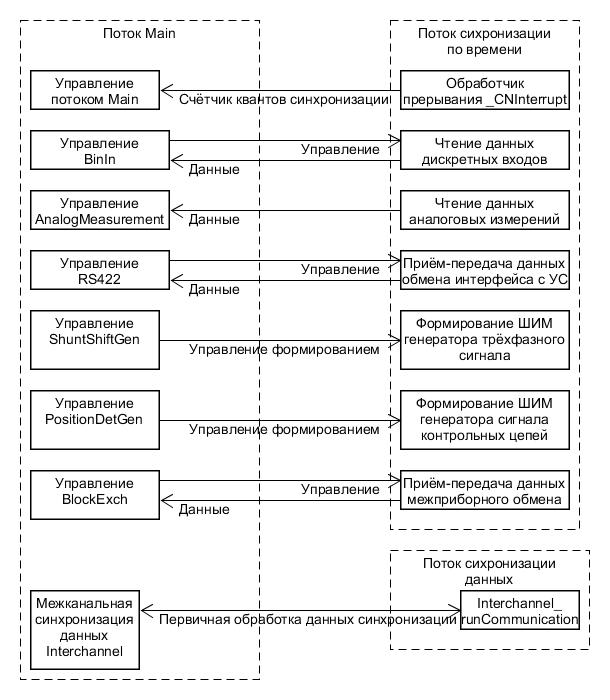


Рисунок 8 – Взаимодействие ГП и потоков синхронизации

* + 1. Взаимодействие режимов

Согласно документу «Функциональные требования к ПО» (см. 3.1.1.1), в архитектуре диаграмма состояний режимов работы ПО (см. Рисунок 8) реализована следующим образом:

1. режим «Состояние настройки периферии и начальной синхронизации»:
2. состояние инициализации перифиерии прибора и синхронищации по времени, реализованное в компоненте **Initial**).
3. режим «Состояние инициализации»:
4. состояние инициализации (**StateInitial**).
5. режим «БС»:
6. безопасное пассивное состояние (**StateSafePassive**);
7. безопасное активное состояние (**StateSafeActive**).
8. режим «РС»:
9. пассивное рабочее состояние (**StatePassive**);
10. активное рабочее состояние (**StateActive**).
11. режим «ЗС»:
12. защитное состояние (**ModeProtection**).

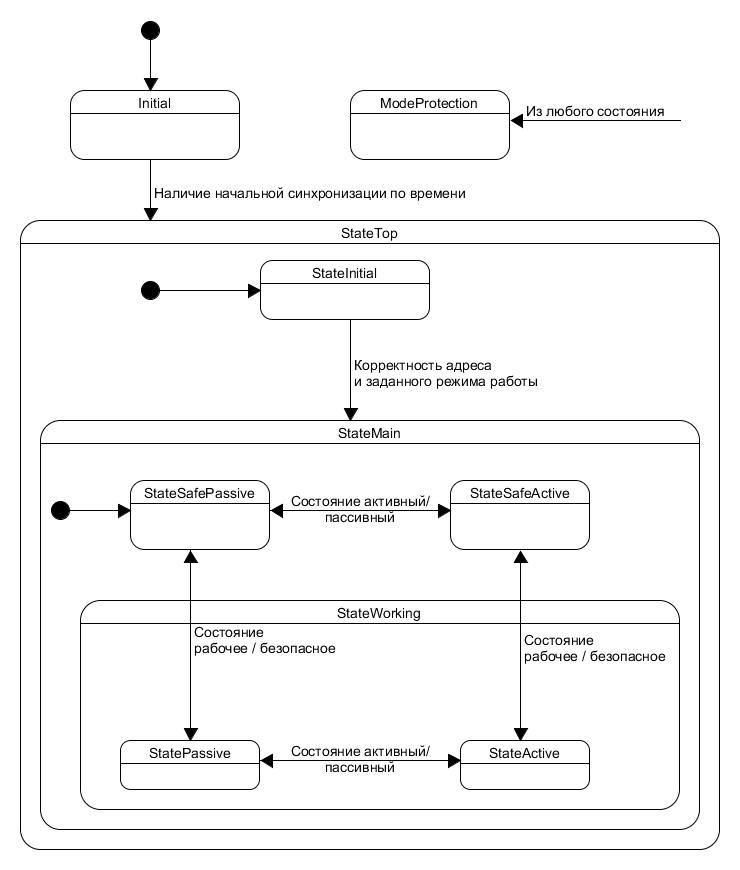


Рисунок 9 – Диаграмма состояний режимов работы ПО

Варианты перехода между режимами:

* состояние начальной инициализации(**Initial**)

Происходит переход в состояние начальной инициализации после подачи сигнала «Сброс/Reset» на МК прибора.

* ЗС (**ModeProtection**).

Условия перехода в ЗС:

* при невыполнении одного из условий: корректность сброса МК, сброс состояния ЗОт при установленной перемычке, передача содержимого энергонезависимой памяти в ВУ, проверка кода ЗС, инициализация переменных и устройств прибора (из **Initial**);
* при отключении периферии в результате неисправности алгоритмов или аппаратной части (из **StateTop**).
* корневое состояние ДКА(**StateTop**).
* состояние инициализации(**StateInitial**).

Происходит переход в состояние инициализации при выполнении всех условий: корректность сброса МК, сброс состояния ЗОт при установленной перемычке, положительный результат проверки кода ЗС на его отсутствие, инициализация переменных и устройств прибора (из **Initial**).

* состояние основной работы(**StateMain**).

Происходит переход в состояние основной работы при установлении корректной конфигурации устройства (из **StateInitial**).

* безопасное пассивное состояние (**StateSafePassive**).

Условия перехода в безопасное пассивное состояние:

* при установлении корректности адреса устройства (из **StateInitial**);
* при невыполнении условий РС (из **StateWorking**);
* при потере активности (из **StateSafeActive**);
* безопасное активное состояние (**StateSafeActive**).

Условия перехода в безопасное активное состояние:

* при установлении корректности адреса устройства (из **StateInitial**);
* при невыполнении условий РС (из **StateWorking**);
* при получении активности (из **StateSafePassive**).
* РС (**StateWorking**).
* пассивное состояние (**StatePassive**).

Условия перехода в пассивное состояние:

* при выполнении условий РС (из **StateSafePassive**);
* при потере активности (из **StateActive**).
* активное состояние (**StateАсtive**).

Условия перехода в активное состояние:

* при выполнении условий РС (из **StateSafeActive**);
* при получении активности (из **StatePassive**).
  + 1. Взаимодействие компонентов

Информационное взаимодействие компонентов показано на рисунке 9.

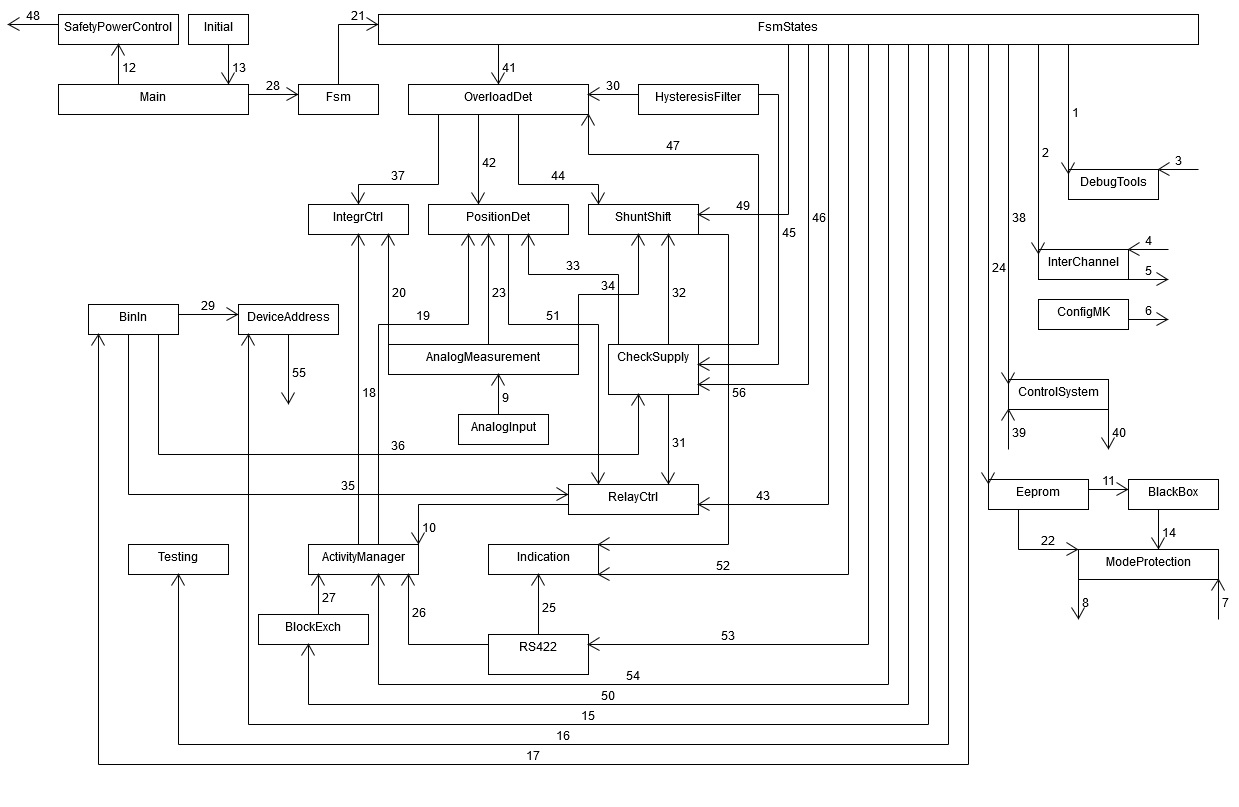


Рисунок – Информационное взаимодействие компонентов

На рисунке 9 представлены следующие информационные связи между компонентами и некоторыми модулями:

1. – управление работой компонента *DebugTools*;
2. – управление работой компонента *InterСhannel*;
3. – данные для модуля *DebugTools*, выводимые по технологическому интерфейсу SPI;
4. – данные, получаемые модулем *InterСhannel* от других модулей для их синхронизации;
5. – синхронизированные данные, полученные в модуле InterСhannel для их использования в других модулях;
6. – данные о типе МК (“Master” или ”Slave”);
7. – данные от любого из модулей при переходе прибора в ЗС (на рисунке не показаны, чтобы не перегружать информацией);
8. – выполнение перехода в ЗС;
9. – результаты опроса каналов АЦП для работы модуля аналогових измерений *AnalogMeasurement*;
10. – данные о состоянии реле РПВ для *ActivityManager*;
11. – сохранённые в ЕЕPROM данные для чтения “чёрного ящика”;
12. – управление источником безопаcного питания *SafetyPowerControl* ;
13. – выполнение начальной синхронизации МК;
14. – данные чёрного ящика для записи в ЕЕPROM при уходе в ЗС.
15. – управление модулем *DeviceAddress*;
16. – управление модулем *Testing*;
17. – управление модулем *BinIn*;
18. – данные об активности прибора от *ActivityManager* к *IntegrCtrl*;
19. – данные об активности прибора от *ActivityManager* к *PositionDet*;
20. – данные аналоговых измерений фазных токов и напряжений при тестировании целостности рабочих цепей для *IntegrCtrl*;
21. – управление состояниями работы прибора;
22. – данные о коде ЗС из EEPROM;
23. – данные аналоговых измерений датчиков положения стрелки для *PositionDet*;
24. – управление работой модуля EEPROM;
25. – данные о поступивших приказах от УС для модуля *Indication*;
26. – вычисление контрольной суммы для обмена между основным и резервным прибором;
27. – результаты обмена между основным и резервным приборами для *ActivityManager*;
28. – управление диспетчером задач;
29. – данные дискретных входов для вычисления адреса, конфигурации прибора в *DeviceAddress*;
30. – результат работы антидребезгового фильтра логических сигналов для сигналов короткого замыкания робочих и контрольних цепей;
31. – данные о наличии питающих напряжений для *ReleayCtrl*;
32. –данные о наличии питающих напряжений для *ShuntShift*;
33. – данные о наличии питающих напряжений для *PositionDet*;
34. – данные аналоговых измерений фазных токов во время перевода стрелки для *ShuntShift*;
35. – данные о состоянии реле РПВ для *RelayCtrl*;
36. – данные о состоянии дискретных входов определения питаний 24 В и 220 В для *CheckSupply*;
37. – данные о налички короткого замыкания робочих цепей для *IntegrCtrl*;
38. – управление работой модуля управляющей логики *ControlSystem*;
39. – данные от компонентов *ShuntShift*, *RS422*, *PositionDet*, *OverloadDet* и др. для построения логики работы *ControlSystem*;
40. – обмен с УС (формирование статусов, алармов), управление активностью прибора, управление индикацией, обработка поступивших от управляющей системы приказов и формирование статусов;
41. – управление работой *OverloadDet*;
42. – данные о коротком замыкании контрольных цепей для *PositionDet*;
43. – управление работой *RelayCtrl*;
44. – данные о КЗ рабочих цепей либо данные об аппаратной неисправности датчика короткого замыкания рабочих цепей для *ShuntShift*;
45. – результат работы антидребезгового фильтра логических сигналов для сигналов наличия питания 24 В и 220 В;
46. – управление работой *CheckSupply*;
47. – данные о наличии 24 В и 220 В для *OverloadDet*;
48. – данные для перехода в ЗС в случае ненормы безопасного источника питания *SafetyPowerControl*;
49. – управление работой *ShuntShift*;
50. – управление работой модуля *BlockExch*;
51. – данные о положении стрелки для *RelayCtrl*;
52. – управление *Indication*;
53. – управление *RS422*;
54. – управление *ActivityManager*;
55. – данные об адресе, конфигурации прибора для *PositionDet*, *ShuntShift*, *ActivityManager*, *RelayCtrl*, *ControlSyste*;
56. – данные модуля *ShuntShift* для индикации светодиодов положения стрелки во время перевода стрелки;
    1. Интерфейсы взаимодействия
       1. Компоненты
          1. Компонент ActivityManager
             1. Назначение

Управление активностью прибора.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* ActivityManager – управление активностью прибора;
* ActivityManager\_connect – обеспечение взаимодействия ActivityManager и BlockExch;
* ActivityManager\_dataTypes –типы данных для модуля управления активностью прибора;
* ActivityManager\_internal – внутренние (неинтерфейсные) функции ActivityManager;
* BlockExch – модуль обмена и связи между основным и резервным приборами.
  + - * 1. Интерфейсы компонента
* ActivityManager\_DeviceStr\* ActivityManager\_ctor()

Инициализация.

* bool ActivityManager\_getNeighborConnection()

Получает состояние связи с соседом.

* bool ActivityManager\_getRS422neighborCtrlSysConnect ()

Получает состояние связи соседа c УС.

* bool ActivityManager\_isActive()

Получает состояние активности прибора.

* bool ActivityManager\_isNeighborWorking()

Получает состояние соседа.

* bool ActivityManager\_isRelayCtrlOn()

Получает состояние управления реле.

* void ActivityManager\_run()

Управление работой модуля.

* void ActivityManager\_setActiveCommand()

Установка наличия активной команды.

* void ActivityManager\_setEnable()

Включение / выключение модуля.

* void ActivityManager\_setRPVoutState()

Установка состояния контактов реле РПВ.

* void ActivityManager\_setRS422connect()

Установка состояния связи по интерфейсу RS-422

* *void ActivityManager\_setRS422ctrlSysConnect()*

Установка признака наличия обмена с УС по интерфейсу RS422

* void ActivityManager\_setState()

Установка состояния прибора.

* void ActivityManager\_changeActivity()

Cмена активности по приказу от УС.

* bool ActivityManager\_getInitDeviceState()

Получить состояние инициализации менеджера активности cвоего прибора.

* void ActiveManagerConnect\_run()

Управление взаимодействием модулей ActivityManager и BlockExch.

* void ActivityManager\_setNeighborConnection()

Установление состояния связи с соседним прибором

* void ActivityManager\_setNeighborHearsMe()

Установка состояния соседнего прибора и прием данных от "моего" прибора.

* void ActivityManager\_setNeighborState()

Установка состояния соседа (рабочее / безопасное).

* ActivityManager\_setNeighborActivity()

Установка состояния активности соседа (активный / пассивный).

* void ActivityManager\_setNeighborRS422connect()

Установка состояния связи соседа по интерфейсу RS-422.

* bool ActivityManager\_getState()

Получает состояние прибора.

* bool ActivityManager\_getRS422connect()

Получает состояние связи по интерфейсу RS-422.

* void ActivityManager\_writeNeighborIsInWork()

Запись состояния инициализации соседа в структуру состояния менеджера активности.

* BlockExch\_Str \*BlockExch\_ctor()

Инициализация модуля обмена между основным и резервным приборами.

* void BlockExch\_run()

Управление приемом и передачей данных межприборного обмена.

* void BlockExch\_interrupt()

Обработка прерывания.

* void BlockExch\_setData()

Формирование данных для передачи.

* bool BlockExch\_getData()

Получает принятые данные от соседнего прибора.

* bool BlockExch\_getConnectionState()

Получает состояние связи с соседним прибором.

* + - 1. Компонент AnalogInput
         1. Назначение

Обработка аналоговых сигналов.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* AnalogInput – модуль обработки аналоговых сигналов;
* Adc – модуль работы с АЦП, МК dsPIC33*;*
* DigitalFilter – модуль цифрового фильтра
* *dsPIC33\_FIR\_filter* – КИХ фильтр для dsPIC33.
  + - * 1. Интерфейсы компонента
* void AnalogInput\_ctor()

Инициализация компонента.

* const AnalogInputValue\* AnalogInput\_getData()

Чтение измеренных значений.

* void AnalogInput\_run()

Обработка аналоговых сигналов.

* void AnalogInput\_setChannelSettings()

Настройка измерительного канала.

* void AnalogInput\_runInterrupt()

Управление дискретизацией сигналов.

* void Adc\_ctor()

Инициализация модуля работы с АЦП.

* void Adc\_run()

Управление драйвером АЦП.

* void Adc\_get()

Чтение результатов из очереди измерений

* bool Adc\_isReady()

Проверка наличия результатов в очереди измерений

* void DigitalFilter\_run()

Фильтрация данных.

* void DigitalFilter\_setFreq()

Настраивает цифрового фильтра на заданную АЧХ.

* + - 1. Компонент AnalogMeasurement
         1. Назначение

Оцифровка, фильтрация, вычисление аналоговых параметров и приведение их к физическим величинам.

* + - * 1. Состав компонента

Компонент состоит из одного одноименного модуля

* + - * 1. Интерфейсы компонента
* void AnalogMeasurement\_ctor()

Инициализация переменных компонента.

* void AnalogMeasurement\_run()

Обработка аналоговых сигналов.

* AnalogMeasurement\_runInterrupt()

Чтение результатов измерения АЦП.

* const AnalogMeasurementValue \*AnalogMeasurement\_getData()

Чтение измеренных значений (определяется используемой системой физических величин).

* + - 1. Компонент BinIn
         1. Назначение

Чтение дискретных входов.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* BinIn – обработка чтения дискретных входов;
* BinInDecoder – чтение дискретных входов, подключенных через дешифратор.
  + - * 1. Интерфейсы компонента
* void BinIn\_ctor()

Инициализация переменных компонента.

* uint16\_t BinIn\_getAddrJumpers()

Возвращает состояние перемычек адреса.

* uint8\_t BinIn\_getCrcJumpers()

Возвращает состояние перемычек контрольной суммы CRC-8.

* uint8\_t BinIn\_getConfigJumpers()

Возвращает состояние перемычек конфигурации CONF7…CONF0.

* bool BinIn\_is220vOk()

Возвращает синхронизированный результат контроля напряжения источника питания 220 В.

* void BinIn\_interruptDataRead()

Чтение тетрады данных.

* void BinIn\_interruptAddrSet()

Установка адресного дешифратора чтения данных в прерывании.

* bool BinIn\_is24VOk()

Возвращает результат контроля напряжения источника питания 24 В.

* bool BinIn\_isRpvOnMe()

Возвращает состояние контактов реле РПВ.

* void BinIn\_run()

Обработка входных сигналов, считанных в прерываниях

функцией BinIn\_interruptDataRead().

* + - 1. Компонент BlackBox
         1. Назначение

Формирование и чтение информации «чёрного ящика».

* + - * 1. Состав компонента

Компонент состоит из одного одноименного модуля.

* + - * 1. Интерфейсы компонента
* void BlackBox\_save()

Сохранить в EEPROM информацию «чёрного ящика».

* void BlackBox\_read()

Чтение из EEPROM информации «чёрного ящика» и выдача ее в канал связи.

* void BlackBox\_saveAddCodeProtectionState()

Сохранить параметр, уточняющий причину ЗС.

* void BlackBox\_saveSourceReset()

Сохранить параметр, уточняющий причину запуска МК.

* void BlackBox\_saveASSERT()

Сохранить имя файла и номер строки, вызвавшего ЗС.

* + - 1. Компонент CheckSupply
         1. Назначение

Проверка состояния внешних источников питания.

* + - * 1. Состав компонента

Компонент состоит из одного одноименного модуля.

* + - * 1. Интерфейсы компонента
* void CheckSupply\_ctor()

Инициализация и запуск.

* bool CheckSupply\_is220vOn()

Получает состояние питания 220 В.

* bool CheckSupply\_is24vOn()

Получает состояние питания 24 В.

* void CheckSupply\_run()

Управление работой модуля.

* + - 1. Компонент ConfigMK
         1. Назначение

Определение конфигурации процессора.

* + - * 1. Состав компонента

Компонент состоит из одного одноименного модуля.

* + - * 1. Интерфейсы компонента
* void ConfigMK\_ctor()

Определение типа процессора: Master или Slave.

* bool ConfigMK\_isMaster()

Определение конфигурации процессора как Master.

* bool ConfigMK\_isSlave()

Определение конфигурации процессора как Slave.

* + - 1. Компонент DebugTools
         1. Назначение

Средства отладки прибора и трассировки данных.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* DebugTools – средства отладки прибора;
* Tracing – трассировка данных прибора.
  + - * 1. Интерфейсы компонента
* void DebugTools\_ctor()

Инициализация модуля DebugTools.

* void DebugTools\_trWordSPI()

Передача по SPI слова данных.

* DebugTools\_trBuffSPI()

Передача по SPI массива данных data с размерностью *size.*

* void DebugTools\_testSPI()

Тест отладочного интерфейса SPI (выдача чисел от 0 до 99).

* void Tracing\_ctor()

Инициализация модуля Tracing.

* void Tracing\_saveBlackBox()

Cохранение буферов трассировки в EEPROM.

* Tracing\_parameter()

Запись данных в буфер трассировки.

* Tracing\_parameterId()

Запись данных в буфер трассировки с идентификатором.

* + - 1. Компонент DeviceAddress
         1. Назначение

Определение адреса и конфигурации прибора

* + - * 1. Состав компонента

Компонент состоит из одного одноименного модуля

* + - * 1. Интерфейсы компонента
* void DeviceAddress\_ctor()

Инициализация и запуск модуля.

* void DeviceAddress\_run()

Управление работой.

* uint16\_t DeviceAddress\_getAddress()

Получает адрес прибора.

* bool DeviceAddress\_isPrimary()

Получает статус прибора.

* DeviceAddress\_sysType DeviceAddress\_getSysType()

Получает тип УС.

* bool DeviceAddress\_isValid( void )

Получает флаг корректности адреса.

* RelayModuleType DeviceAddress\_getRelayModuleType()

Получает тип модуля реле.

* MotorType DeviceAddress\_getMotorType()

Получает тип электродвигателя.

* + - 1. Компонент Eeprom
         1. Назначение

Работа с внешним EEPROM.

* + - * 1. Состав компонента

Компонент состоит из одного одноименного модуля

* + - * 1. Интерфейсы компонента
* void Eeprom\_ctor()

Инициализация модуля EEPROM;

* void Eeprom\_run()

Рабочий цикл работы с EEPROM;

* uint16\_t Eeprom\_read()

Чтение из EEPROM 16-битного значения.

* void Eeprom\_write()

Запись в ячейку EEPROM 16-битного значения.

* bool Eeprom\_isReady()

Готовность EEPROM к чтению или записи данных.

* + - 1. Компонент Fsm
         1. Назначение

Иерархический детерменированный конечный автомат.

* + - * 1. Состав компонента

Компонент состоит из одного одноименного модуля.

* + - * 1. Интерфейсы компонента
* void Fsm\_run()

Работа ДКА.

* void Fsm\_start()

Запуск ДКА.

* void Fsm\_transit()

Переход в состояние.

* + - 1. Компонент FsmStates
         1. Назначение

Планировщик заданий и объявление следующих состояний ДКА.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* Sheduler – планировщик заданий;
* StateActive – активное состояние;
* StateWorking – рабочее состояние;
* StateInitial – состояние инициализации;
* StateMain – состояние основной работы;
* StatePassive – пассивное состояние;
* StateSafeActive – безопасное активное состояние;
* StateSafePassive – безопасное пассивное состояние;
* StateTop – корневое состояние ДКА;
* States – объявление состояний ДКА.
  + - * 1. Интерфейсы компонента

Интерфейсы взаимодействия заложены в структуру каждого модуля типа *FsmState* и состоят из следующих указателей на функции:

* void( \*onEntry)()

Обработка входа в состояние.

* void( \*onExit)()

Обработка выхода из состояния.

* bool( \*onRun)()

Обработка событий, проверка условий переходов.

* + - 1. Компонент HysteresisFilter
         1. Назначение

Фильтрация цифровых входов с гистерезисом

* + - * 1. Состав компонента

Компонент состоит из одного одноименного модуля

* + - * 1. Интерфейсы компонента
* void HystFltr\_ctor()

Инициализация фильтруемой цифровой линии в начальное состояние.

* bool HystFltr\_run()

Фильтрация цифрового сигнала.

* + - 1. Компонент Indication
         1. Назначение

Управление индикацией светодиодов

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* LedFailure – управление светодиодом «Отказ»;
* Indication – управление остальніми светодиодами.
  + - * 1. Интерфейсы компонента
* void Indication\_blink()

Мигание индикатором циклически с заданной начальной фазой.

* void Indication\_blink2()

Мигание индикатором циклически с двойным периодом и заданной начальной фазой.

* void Indication\_ctorDummy()

Инициализация конкретного объекта индикатора как заглушки.

* void Indication\_ctorLed()

Инициализация конкретного объекта индикатора.

* void Indication\_off()

Выключение индикатора.

* void Indication\_on()

Включение индикатора.

* void Indication\_pulseOn()

Включение индикатора однократно на заданное время.

* void Indication\_run()

Обработка состояния конкретного объекта индикатора.

* void LedFailure\_ctor()

Инициализация управления светодиодом "Отказ".

* void LedFailure\_set()

Установка состояния светодиода "Отказ".

* void LedFailure\_blink()

Мигание светодиодом "Отказ".

* + - 1. Компонент Initial
         1. Назначение

Начальная инициализация после сброса МК, настройка начальной синхронизации по времени между МК, получение данных о приборе и запись их в EEPROM.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* CheckCauseReset – определение причин сброса МК;
* TimeBeginSynchronization – начальная временная синхронизация МК;
* *IdentHex* – идентификация прибора и его ПО (имя, версия, дата);
* *TimeSynchronizationPorts* – порты для временной синхронизации МК.
  + - * 1. Интерфейсы компонента
* void checkCauseReset\_run()

Определение причин сброса МК;

* void IdentHex\_run()

Запись в EEPROM имени прибора, версии, даты прошивки.

* void TimeBeginSynchronization\_run()

Начальная синхронизация МК по времени.

* + - 1. Компонент IntegrCtrl
         1. Назначение

Контроль целостности обмоток электродвигателя.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* *IntegrCtrl* – логика контроля целостности обмоток двигателя;
* *IntegrCtrlGen* – формирователь сигнала контроля целостности обмоток двигателя.
  + - * 1. Интерфейсы компонента
* void IntegrCtrl\_ctor()

Инициализация модуля контроля целостности обмоток двигателя.

* void IntegrCtrl\_run()

Управление работой модуля контроля целостности обмоток.

* void IntegrCtrl\_turnOff()

Отключение модуля контроля целостности обмоток.

* bool IntegrCtrl\_isOn()

Проверка состояния модуля контроля целостности обмоток.

* void IntegrCtrl\_turnOn()

Включение модуля контроля целостности обмоток.

* MotorFailure IntegrCtrl\_getFailure()

Получить значение состояния рабочих цепей.

* void IntegrCtrlGen\_ctor()

Инициализация генератора.

* void IntegrCtrlGen\_run()

Управление работой формирователя сигнала контроля целостности обмоток.

* void IntegrCtrlGen\_turnOn()

Включение генератора контроля целостности обмоток.

* void IntegrCtrlGen\_turnOff()

Отключение формирователя сигнала контроля целостности обмоток.

* void IntegrCtrlGen\_setParam()

Установка параметров генератора.

* + - 1. Компонент InterChannel
         1. Назначение

Межканальный обмен.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* InterChannel – межканальный обмен;
* log2 – определение таблицы Log2\_log2Lkup[];
* pwr2 – определение таблиц Pwr2\_pwr2Lkup[], Pwr2\_invPwr2Lkup[], and Pwr2\_div8Lkup[];
* set – очередь с приоритетом, состоящая из 8 или 64 элементов.
  + - * 1. Интерфейсы компонента
* void InterChannel\_ctor ()

Инициализация переменных компонента.

Примечание

Инициализируются все данные для корректной работы компонента.

Вызов этой функции перед использованием других функций этого компонента обязателен.

* void InterChannel\_setParamSettings ()

Настройка сценариев синхронизации параметра.

* void InterChannel\_synchronize ()

Запрос синхронизации параметра.

* bool InterChannel\_isSynchronized ()

Признак завершения синхронизации параметра.

* bool InterChannel\_syncEvent()

Cобытие синхронизации.

Примечание

Функция сообщает о том, что между текущим и предыдущим ее вызовом.

* bool InterChannel\_syncEventOverflow()

Признак переполнения событий синхронизации.

Примечание

Функция сообщает о том, что с момента последнего чтения события синхронизации, произошло больше одного события.

* bool InterChannel\_isHandling ()

Признак процесса выполнения синхронизации параметра.

* param\_t InterChannel\_getData ()

Чтение синхронизированного значения параметра.

* param\_t InterChannel\_getLocalData ()

Чтение значения своего канала, которое было отправлено на синхронизацию.

* void InterChannel\_run()

Синхронизация параметров.

Примечание

Основной поток. Функция выполняет контроль времен Т1, Т2 и Т3, выполняет обработку ресурсоемких частей алгоритма, а также вызывается из главного цикла программы с периодом 1 мс.

* void InterChannel\_runCommunication ()

Работа с соседним каналом через линию связи.

Примечание

Определяет события готовности данных к чтению или передаче, выполняет чтение данных из драйвера и инициирует передачу. Функция должна вызываться из подпрограммы обработки прерываний синхронизации. Частота вызова в соответствии с частотой этих прерываний.

* void InterChannel\_runDrv()

Опрос готовноcти драйвера.

* void InterChannel\_setCommunicationData()

Установка значения параметра в физическом протоколе передачи данных.

* void InterChannel\_getCommunicationData()

Получает значение параметра в физическом протоколе.

* + - 1. Компонент Main
         1. Назначение

Основной цикл, обработчик прерываний.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* ControlSystem – общее управление работой прибора;
* Main – основной цикл, обработчик прерываний и инициализация;
* InterruptsHandlers – обработка прерываний;
* UnusedInterrupts – обработка неиспользуемых прерываний.

Примечание

Вызов функций инициализации всех компонентов осуществляется в функции Main перед основным циклом;

* + - * 1. Интерфейсы компонента
* uint16\_t Main\_getTimeWorkInterrupt()

Получить количество выполненных прерываний (период 62,5 мкс) из основного цикла.

* uint16\_t Main\_getTimeWorkMs()

Получить время работы внутри секунды, мс.

* uint16\_t Main\_getTimeWorkSec()

Получить время работы, с.

* void ControlSystem\_ctor()

Инициализация модуля общего управления.

* void ControlSystem\_run()

Работа модуля общего управления.

* bool ControlSystem\_workCondition()

Проверка условий рабочего режима.

* + - 1. Компонент ModeProtection
         1. Назначение

Режим «Защитное состояние»

* + - * 1. Состав компонента

Компонент состоит из одноимённого модуля.

* + - * 1. Интерфейсы компонента
* void ModeProtection\_ctor()

Настройка передачи кода защитного отказа (используется в отладочном режиме);

* void ModeProtection\_run()

Переход в защитное состояние.

Примечание

При отключении периферии происходит запись в EEPROM кода ЗС и информации чёрного ящика.

* void ModeProtectiont\_transmitToWait()

Переход в состояние бесконечного.

* void ModeProtection\_infinityLoop()

Пустой бесконечный цикл;

* void ModeProtection\_ClearPS()

Снятие состояния защитного отказа.

* + - 1. Компонент OverloadDet
         1. Назначение

Определение перегрузок по току.

* + - * 1. Состав компонента

Компонент состоит из одноимённого модуля.

* + - * 1. Интерфейсы компонента
* void OverloadDet\_ctor()

Инициализация компонента.

* void OverloadDet\_run()

Управление работой компонента.

* bool OverloadDet\_isScThreePhaseGen()

Проверка наличия кроткого замыкания в цепях 3-фазного генератора.

* bool OverloadDet\_igetScThreePhaseGenTrig()

Проверка флага КЗ, который идёт в поле статуса при обмене с УС.

* OverloadDet\_isScContrGen()

Проверка наличия КЗ в цепях контрольного генератора.

* bool OverloadDet\_igetScThreePhaseGenTrig ()

Проверка наличия срабатывания триггера КЗ в цепях контрольного генератора.

* + - 1. Компонент PositionDet
         1. Назначение

Определение положения стрелки и генерация сигнала контрольных цепей.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* posDetGenerator – генератор напряжения контрольных цепей;
* positionDet – определение положения стрелки;
* *positionDetFiveEC* – определение положения c модулем реле 5-ЭЦ;
* *positionDetNineWire* – определение положения c модулем реле 9-ти проводным;
* *positionDetTypes* – прототипы функций определения положения.
  + - * 1. Интерфейсы компонента
* void PosDetGenerator\_ctor()

Инициализация генератора напряжения контрольных цепей.

* void PosDetGenerator\_run()

Управление работой генератора напряжения контрольных цепей в основном цикле.

* void PosDetGenerator\_interrupt()

Управление работой генератора напряжения контрольных цепей в прерывании.

* void PosDetGenerator\_turnOn()

Включение/выключение генератора напряжения контрольных цепей.

* bool PosDetGenerator\_isEnable()

Проверка состояния генератора напряжения контрольных цепей.

* PosDetGenerator\_checkUfb()

Контроль напряжения обратной связи генератора напряжения контрольных цепей.

* *uint8\_t PosDetGenerator\_isGeneratorFail()*

Возвращает состояние генератора напряжения контрольных цепей.

* void PosDetGenerator\_setLevel ()

Установка уровня генератора напряжения контрольных цепей.

* uint8\_t PosDetGenerator\_getLevel()

Получить уровень генератора напряжения контрольных цепей.

* uint8\_t PosDetGenerator\_isReadyForUse()

Получить состояние готовности генератора напряжения контрольных цепей к выдаче достоверных данных.

* void PositionDet\_ctor()

Инициализация определения положения стрелки.

* void PositionDet\_run()

Управление работой определения положения стрелки.

* void PositionDet\_turnOn()

Включение/выключение определения положения.

* bool PositionDet\_isEnable()

Проверка состояния модуля определения положення.

* PositionDet\_State PositionDet\_getPosition()

Получает текущее положение стрелки.

* PositionDet\_State PositionDet\_getRequestPositionDet()

Получает запрос на установку позиции контроля положения.

* void PositionDet\_setPosDet()

Передает установленную позицию контроля.

* + - 1. Компонент RelayCtrl
         1. Назначение

Управление реле и контроль состояния контактов реле РПВ.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* relayCtrl – управление реле и контроль состояния контактов РПВ;
* relayCtrlFiveEC – управление модулем реле 5-ЭЦ;
* *relayCtrlNineWire* – управление модулем реле 9-ти проводным;
* *relayCtrlTypes* – прототипы функций управления и контроля модуля реле.
  + - * 1. Интерфейсы компонента
* void RelayCtrl\_ctor()

Инициализация и запуск.

* void RelayCtrl\_run()

Управление работой;

* void RelayCtrl\_setShiftDir()

Установить направление перевода стрелки.

* void RelayCtrl\_setPosDet()

Установить контроль положення стрелки.

* ShiftDirection RelayCtrl\_getShiftDir()

Получает установленное направление перевода стрелки.

* PositionDet\_State RelayCtrl\_getPosDet()

Получает контролируемое положение стрелки.

* void RelayCtrl\_switchRpv()

Включение/выключение реле РПВ.

* RelayCtrl\_RpvState RelayCtrl\_getRpvState()

Получает состояние исправности реле РПВ.

* + - 1. Компонент RS422
         1. Назначение

Обеспечение связи с управляющей системой по интерфейсу RS-422.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* Rs422 – обеспечение связи с УС по интерфейсу RS-422;
* Rs422\_crc8 – вычисление контрольной суммы принятого пакета данных;
* Rs422\_dataTypes *–* модуль RS-422. Внутренние типы данных;
* Rs422\_lineExch – организация обмена по линии RS-422;
* Rs422\_transceiver – внутренние функции обмена по интерфейсу  
  RS-422.
  + - * 1. Интерфейсы компонента
* Rs422\_handler\* Rs422\_ctor()

Инициализация.

* void Rs422\_run()

Управление приемом и передачей данных.

* void Rs422\_interrupt()

Прием и передача данных.

* void Rs422\_setEnable()

Разрешение работы модуля.

* void Rs422\_setStatus()

Установка статуса прибора.

* bool Rs422\_getLineEvent()

Возвращает признак событий обмена на указанной линии связи.

* void Rs422\_addAlarm()

Добавление кода аларма в список актуальных для прибора.

* bool Rs422\_getNoExchangeCtrlSystem()

Возвращает признак отсутствия связи с УС по обеим линиям.

* bool Rs422\_getNoLineExchange()

Возвращает признак отсутствия связи по RS-422 по указанной линии.

* bool Rs422\_getNoLineExchangeCtrlSystem()

Возвращает признак отсутствия связи с УС.

* uint8\_t Rs422\_getOrder()

Обработка приказа и чтение его длины.

* void Rs422\_removeAlarm()

Удаление кода аларма из списка актуальных для прибора.

* bool Rs422\_getNewOrderEventLine()

Признак наличия нового приказа.

* + - 1. Компонент SafetyPowerControl
         1. Назначение

Управление и контроль безопасного источника питания

* + - * 1. Состав компонента

Состоит из одноимённого модуля

* + - * 1. Интерфейсы компонента
* void SafetyPowerControl\_ctor()

Инициализация компонента;

* void SafetyPowerControl\_runInterrupt()

Реализация формирования безопасного питания и его контроля.

* + - 1. Компонент ShuntShift
         1. Назначение

Управление переводом стрелки

* + - * 1. Состав компонента

Компонент состоит из следующих модулей:

* *shuntShift –* управление переводом стрелки;
* *shuntShiftFiveEC –* управления переводом c модулем реле 5-ЭЦ;
* *shuntShiftGen –* генерация 3-фазной синусоиды с заданной частотой, уровнем и фазировкой;
* *shuntShiftMotor –* управление работой двигателя;
* *shuntShiftNineWire –* управления переводом c модулем реле 9-ти проводным;
* *shuntShiftTypes –* прототипы функций управления переводом.
  + - * 1. Интерфейсы компонента
* void ShuntShift\_ctor()

Инициализация модуля перевода стрелки.

* void ShuntShift\_run()

Управление работой модуля перевода стрелки.

* void ShuntShift\_turnOn()

Включение перевода стрелки.

* void ShuntShift\_turnOff()

Выключение перевода стрелки.

* void ShuntShift\_setDetPos()

Передача положения стрелки.

* void ShuntShift\_setShiftPos()

Передача установленной позиции перевода.

* bool ShuntShift\_isEnable()

Проверка состояния перевода стрелки.

* PositionDet\_State ShuntShift\_getReqPosDet()

Запрос на установку позиции контроля положення.

* ShiftDirection ShuntShift\_getReqShiftDir()

Запрос на установку позиции направления перевода стрелки.

* uint16\_t ShuntShift\_getTime()

Получить время последнего перевода.

* ShiftDgn ShuntShift\_getDgn()

Получить значение поля статуса dgn.

* void ShuntShift\_setCntPhaseShift()

Установить значение счётчика фазы стрелочного перевода.

* const uint8\_t ShuntShift\_getCntPhaseShift()

Получить значение счётчика фазы стрелочного перевода.

* void ShuntShiftGen\_ctor()

Инициализация генератора трёхфазного сигнала.

* void ShuntShiftGen\_run()

Управление работой генератора трёхфазного сигнала в основном цикле

* void ShuntShiftGen\_interrupt()

Управление работой генератора трёхфазного сигнала в прерывании.

* void ShuntShiftGen\_turnOff()

Отключение генератора трёхфазного сигнала.

* void ShuntShiftGen\_fastTurnOff()

Быстрое отключение генератора трёхфазного сигнала при коротком замыкании рабочих цепей.

* bool ShuntShiftGen\_isOn()

Получить состояние работы генератора.

* void ShuntShiftGen\_turnOn()

Включение генератора трёхфазного сигнала.

* void ShuntShiftGen\_setParam()

Установить параметры сигнала (частота и уровень).

* void ShuntShiftMotor\_ctor()

Инициализация модуля управления двигателем.

* void ShuntShiftMotor\_run()

Управление работой модуля управления двигателем.

* void ShuntShiftMotor\_turnOn()

Включение двигателя.

* void ShuntShiftMotor\_turnOff()

Выключение двигателя.

* bool ShuntShiftMotor\_isOn()

Проверка состояния модуля.

* MotorFailure ShuntShiftMotor\_getFailure()

Получить причину неисправности.

* void ShuntShiftMotor\_setFailure()

Установить неисправность рабочих цепей.

* uint16\_t ShuntShiftMotor\_getWorkingTime

Получить время работы двигателя.

* + - 1. Компонент Testing
         1. Назначение

Проверка работоспособности МК.

* + - * 1. Состав компонента

В состав компонента входят следующие модули:

* ControlMK – проверка работоспособности МК;
* CheckCallFunctions – контроль вызова функций;
* CheckRAM – проверка ОЗУ;
* CheckRegisters – проверка основных регистров МК, используемых при работе;
* FlashCheck – проверка памяти программ, области конфигурации и контрольной суммы памяти программ;
* CheckCPU – проверка ядра МК.
  + - * 1. Интерфейсы компонента
* void CheckCallFunctions\_ctor()

Инициализация модуля контроля вызова функцій.

* void CheckCallFunctions\_run()

Контроль количества вызванных функций.

* void CheckCallFunctions\_interrupt()

Передача для контроля количества вызванных функцій.

* void CheckCallFunctions\_marked()

Маркирование выполнения функции.

* void CheckCallFunctions\_call()

Вызов функции с сохранением ее имени.

* uint16\_t CheckCallFunctions\_saveBlackBox()

Запись последних N\_STACK\_NAME\_FUNCTION функций в EEPROM.

* bool CheckCPU\_run()

Проверка ядра МК.

* void CheckRAM\_ctor()

Инициализация переменных модуля проверки ОЗУ.

* bool CheckRAM\_run()

Проверка ОЗУ.

* bool CheckRegisters\_run()

Проверка основных регистров МК.

* void ControlMK\_ctor()

Инициализация переменных модуля ControlMK.

* void ControlMK\_run()

Проверка работоспособности МК.

* void FlashCheck\_ctor()

Инициализация переменных модуля FlashCheck.

* void FlashCheck\_run()

Проверка памяти программ.

* + 1. Системные каталоги
       1. Каталог drv
          1. Назначение

Набор программных модулей, которые обеспечивают доступ к периферии МК через интерфейсные функции.

* + - * 1. Состав каталога

В состав каталога входят следующие модули:

* AdcDriver\_PIC33 *–* драйвер АЦП для МК dsPIC33;
* *Adc\_Inic1* *–* макроопределения для инициализации аппаратного модуля МК ADC1;
* *Adc\_regsPIC33* *–* регистры модуля Adc;
* BlockExch\_driver – драйвер обмена между основным и резервным приборами;
* *checkRegistersDrv* – проверка основных аппаратных регистров, используемых при работе МК;
* *ConfigurationMK –* конфигурация МК;
* dsPIC33\_rs422 – драйвер модуля RS-422 для МК dsPIC33;
* *ECAN1\_PinRemap –* настройка пинов на работу с ECAN1;
* *ECAN2\_PinRemap –* настройка пинов на работу с ECAN2;
* EcanAck\_driverPIC33 – драйвер для модуля CAN c дополнительной функцией подтверждения приема по дискретным линиям;
* *EcanAck\_Pins* – порты ввода-вывода модуля "Драйвер ECAN с подтверждением приема по дискретным линиям";
* *Ecan\_DMA* – макросы для работы с прямым доступом в память устройств ECAN;
* Ecan\_driverPIC33 – драйвер модуля ECAN для МК dsPIC33;
* *Ecan\_regsPIC33* – порты модуля ECAN;
* EepromDrv – драйвер работы с внешним EEPROM, МК dsPIC33 (подключение - шина I2C);
* *I2Cdrv\_PIC33* – драйвер модуля I2C (Master) для МК dsPIC33;
* iodrv – интерфейсы драйверов;
* *shuntShiftGenDrv –* драйвер генератора трёхфазного напряжения рабочих цепей;
* *posDetGen\_drv –* драйвер генератора напряжения контрольных цепей;
* *I2Cdrv\_PIC33* - драйвер модуля I2C (Master) для МК dsPIC33;
* *IOports -* макросы для работы с портами ввода/вывода;
* MainRegisters – настройка битов конфигурации семейства МК dsPIC33;
* relayCtrlDrv – драйвер управления реле, находящихся на модуле реле.
* *turnOffdsPIC33 –* отключение периферии МК dsPIC33;
* *typeMK –* определение типа МК;
* *WorkWithBits –* макросы для работы с битами.
  + - 1. Каталог systems
         1. Назначение

Каталог systems содержит программные модули, которые используются компонентами ПО для их настройки в конкретном проекте.

* + - * 1. Состав

В состав каталога входят следующие модули:

* addressVarEeeprom – адреса переменных и констант, хранящихся в EEPROM;
* AnalogInputId – идентификаторы входных аналоговых сигналов;
* *asserts* – управляемые утверждения;
* asserts\_ex – расширенные утверждения;
* defCompil – определение условий компилятора;
* *InterChannelId* – идентификаторы параметров межпроцессорного обмена;
* *nameHex* – имя прибора;
* *ProtectionState\_codes* – коды отказов ОКПС-Е-К;
* *version* – версия и дата сборки ПО.
  + - 1. Каталог Utility
         1. Назначение

Каталог Utility содержит программные модули, которые выполняют задачи общего назначения и не являются фундаментальными с точки зрения построения ПО.

* + - * 1. Состав

В состав каталога входят следующие модули:

* *CopySwap* – копирование массива с переменой байт;
* *SqrtTOpt* – вычисление квадратного корня;
* *wait* – функции программных задержек.

1. Обеспечение безопасности в ПО

Компоненты ПО обеспечивают реализацию функций безопасности, изложенных в 3.2.3 документа «Объектный контроллер привода стрелки. ОКПС-Е-К. Функциональные требования к ПО ЕИУС.хххххх.ххх 01 90 01 01»

При возникновении внешних неблагоприятных условий прибор переходит в БС (см. таблицу 2).

В ходе работы прибор выполняет самодиагностику состояния и переход в ЗС в случае обнаружения аппаратных или программных отказов (см. таблицу 3).

При реализации функции межканальной синхронизация работы программ по данным выполняется обмен данными в соответствии с таблицей 4.

Таблица 1 – Функции безопасности ПО и компоненты верхнего уровня, обеспечивающие их выполнение

| **№** | **Функция безопасности**  **(см. 3.2.3 Функциональных требований к ПО)** | **Компонент ПО** | **Модуль ПО** |
| --- | --- | --- | --- |
|  | Обмен по двум линиям RS-422 в соответствии с Протоколом 1) | Rs422 | Все модули компонента |
|  | Межканальная (Master-Slave) синхронизация работы программ по времени | Main | main |
|  | Межканальная (Master-Slave) синхронизация работы программ по данным | InterChannel | Все модули компонента |
| Main | main |
|  | Контроль наличия связи с УС по RS-422 | Rs422 | Rs422,  Rs422\_lineExch |
|  | Контроль входных напряжений питания «24 В» и «220 В» | CheckSupply | CheckSupply |
|  | Контроль состояния реле в модуле реле АУКС-9 | RelayCtrl | relayCtrl |
|  | Контроль наличия корректного модуля реле, подключаемого к кросс-плате | ShuntShift | shuntShift |
| RelayCtrl | relayCtrl |
| PositionDet | positionDet |
|  | Контроль параметров формируемого сигнала контрольных цепей | PositionDet | posDetGenerator |
|  | Контроль безопасного источника питания | SafetyPowerControl | SafetyPowerControl |
|  | Безопасное считывание данных, задаваемых перемычками на кросс-плате, и контроль их целостности с помощью циклического избыточного кода CRC8 | BinIn | BinIn |
| DeviceAddress | deviceAddress |
|  | Самодиагностика состояния аппаратных средств и проверка корректности работы ПО | См. таблицу 3 | |
|  | Переход в БС при обнаружении внешних негативных условий, при которых невозможно гарантировать безопасную работу устройства | См. таблицу 2 | |
|  | Переход в ЗС при обнаружении внутренних аппаратных или программных отказов, при которых невозможно гарантировать безопасную работу устройства | См. таблицу 3 | |
|  | Для перевода прибора в ЗС должно быть достаточно управления от одного МК | ModeProtection | ModeProtection |
|  | Прибор должен иметь возможность находиться в ЗС неограниченное количество времени и ни при каких обстоятельствах не должен выходить из него самостоятельно | ModeProtection | ModeProtection |
|  | Выход прибора из ЗС должен быть возможен только при вмешательстве обслуживающего персонала и при выполнении им специальных технологических операций | ModeProtection | ModeProtection |
| 1) Протокол – документ «». | | | | |

Таблица 2 – Условия перехода в БС и компоненты ПО, их определяющие

| **№** | **Условие перехода в БС**  **(см. 3.1.1.1.4 Функциональных требований к ПО)** | **Компонент ПО** |
| --- | --- | --- |
|  | Отсутствие связи с УС по RS-422 (отсутствие корректных приказов по двум линиям связи более 1500 мс) | Rs422 |
|  | Не норма (пониженное) входное напряжения питания «24 В» | CheckSupply |
|  | Не норма (пониженное) входное напряжения питания «220 В» | CheckSupply |
|  | Основной прибор: неисправность РПВ по включению | RelayCtrl |

Таблица 3 – Условия перехода в ЗС и компоненты ПО, их определяющие

| **№** | **Условие перехода в ЗС**  **(см. 3.1.1.1.5 Функциональных требований к ПО)** | **Компонент ПО** 1) | **Модуль ПО** 1) |
| --- | --- | --- | --- |
|  | При инициализации прибора, если в EEPROM обнаружены записи с кодом ЗО | Main | main |
|  | Отказ межканальной синхронизации по времени параметров из любых компонентов ПО | Main | main |
|  | Отказ межканальной синхронизации по данным | InterChannel | InterChannel |
|  | Не норма напряжения питания МК «3.3 В» | AnalogMeasurement | AnalogMeasurement |
|  | Не норма опорного напряжения для АЦП соседнего канала | AnalogMeasurement | AnalogMeasurement |
|  | Неисправность корректора коэффициента мощности (не норма напряжения на выходе корректора) | IntegrCtrl | IntegrCtrl |
|  | Неисправность датчика перегрузки по току в рабочих цепях | OverloadDet | OverloadDet |
|  | Неисправность датчика перегрузки по току в контрольных цепях | OverloadDet | OverloadDet |
|  | Неисправность схемы формирования напряжения контрольных цепей (по среднеквадратическому значению напряжения обратной связи) | PositionDet | posDetGenerator |
|  | Неисправность датчиков тока рабочих цепей | AnalogMeasurement | AnalogMeasurement |
|  | Неисправность датчиков напряжения контроля положения стрелки | AnalogMeasurement | AnalogMeasurement |
|  | Неисправность схемы управления реле РПВ | RelayCtrl | relayCtrl |
|  | Неисправность блока чтения состояния перемычек на кросс-плате | BinIn | BinIn |
| DeviceAddress | deviceAddress |
|  | Ошибки при чтении-записи в(из) внешнюю память EEPROM | Eeprom | Eeprom |
|  | Неисправность выходного каскада по результатам измерения параметров формируемого сигнала – при контроле амплитуды выходного сигнала | AnalogInput | AnalogInput |
|  | Неисправность выходного каскада по результатам измерения параметров формируемого сигнала – при контроле фазы выходного сигнала | AnalogInput | AnalogInput |
|  | Несовпадение состояния перемычек на кросс-плате, обнаруживаемое в процессе работы в течение определенного времени, по сравнению с состоянием перемычек, считанным после сброса МК | DeviceAddress | deviceAddress |
|  | Внутренние отказы МК:  Отказ АЦП | AnalogMeasurement | AnalogMeasurement |
| drv | AdсDriver\_PIC33 |
|  | Внутренние отказы МК:  Отказ памяти программ (ПЗУ) | Testing | FlashCheck |
|  | Внутренние отказы МК:  Отказ оперативной памяти данных (ОЗУ) | Testing | CheckRAM |
|  | Внутренние отказы МК:  Отказ АЛУ, регистров W0-W15, регистров цикла и стека | Testing | CheckCPU |
|  | Внутренние отказы МК:  Отказ регистров | Testing | CheckRegisters 2) |
|  | Внутренние отказы МК:  Превышение числа сбросов, вызванных срабатыванием сторожевого таймера или ПО, в течение определенного промежутка времени | Initial | CheckCauseReset |
|  | Внутренние отказы МК:  Срабатывание неиспользуемых прерываний | Main | unused\_interrupt |
|  | Внутренние отказы МК:  Зависание (превышение функциями времени, выделенного на выполнение) | Main | main |
|  | Внутренние отказы МК:  Обнаружение ошибок в процессе контроля вызываемых функций | Все компоненты ПО |  |
| Дополнительные условия перехода в ЗС, не указанные в 3.1.1.1.5 Функциональных требований к ПО | | | |
|  | Ошибка при проверке буферного регистра АЦП | AnalogMeasurement | AnalogMeasurement |
|  | Переполнение счетчика прерываний | Main | Main |
|  | Ошибка уровня выходного сигнала при тестовом импульсе | AnalogInput | AnalogInput |
|  | Ошибки при обмене по линиям связи RS-422 | RS422 | Rs422\_lineExch |
|  | Ошибки при обмене между основным и резервным приборами | ActivityManager | BlockExch |
|  | Ошибка динамического тестирования состояния безопасного источника питания | SafetyPowerControl | SafetyPowerControl |
|  | Одновременный отказ РПВ и несовпадение перемычек на кросс-плате | DeviceAddress | deviceAddress |
| 1) Компоненты/модули ПО, определяющие условия перехода в ЗС, выполняют соответствующие функции самодиагностики.  2) Регистры T1CON и PR5 таймера ТMR5, формирующего кванты синхронизации. | | | |

Таблица 4 – Данные (параметры), синхронизируемые при межканальной синхронизации по данным

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Назначение** | **Сценарий** | **Процедура синхронизации** | **Процедура контроля** | **Время контроля, мс** | | |
| **T1, мс** | **T2, мс** | **T3, мс** |
| **eICId\_Eeprom** | Данные для EEPROM | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 50 | 100 | 0 |
| **eICId\_BlockExchM** | Межблочный обмен, контрольная сумма (свой МК) | eScriptTransmit – передача параметра | eProcSyncOff – без синхронизации | eProcCheckOff – нет контроля | Нет контроля | Нет контроля | Нет контроля |
| **eICId\_BlockExchS** | Межблочный обмен, контрольная сумма (соседний МК) | eScriptTransmit – передача параметра | eProcSyncOff – без синхронизации | eProcCheckOff – нет контроля | Нет контроля | Нет контроля | Нет контроля |
| **eICId\_ControlMK\_CF** | Количество вызванных функций | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 10 | 20 | 128 |
| **eICId\_ControlMK\_Flash** | Контрольная сумма ПЗУ | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 50 | 100 | 45000 |
| **eICId\_Rs422Sync\_Rs1** | Данные по интерфейсу RS1 | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 10 | Нет контроля | Нет контроля |
| **eICId\_Rs422Sync\_Rs2** | Данные по интерфейсу RS2 | eScriptSync – синхронизация параметра | eProcSyncк – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 10 | Нет контроля | Нет контроля |
| **eICId\_OverloadDetScContrGen** | Перегрузка в цепи контрольного генератора (выход логического датчика) | eScriptSync – синхронизация параметра | eProcSyncOR – выбор по ИЛИ | eProcCheckOff – нет контроля | 50 | 200 | Нет контроля |
| **eICId\_OverloadDetScThreePhaseGen** | Перегрузка в цепях управления двигателем (выход логического датчика) | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 50 | 200 | Нет контроля |
| **eICId\_ShuntShiftMotor\_failure 1)** | Состояние рабочих цепей при переводе стрелки | eScriptSync – синхронизация параметра | eProcSyncOR – выбор по ИЛИ | eProcCheckOff – нет контроля | 50 | Нет контроля | Нет контроля |
| **eICId\_IntegrCtrlScAndBreak** | Состояние рабочих цепей при работе алгоритма контроля целостности цепей стрелки | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 100 | 200 | Нет контроля |
| **eICId\_BinIn** | Дискретные входы, считываемые через дешифратор (адрес прибора, конфигурация, контрольная сумма) | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 500 | 500 | Нет контроля |
| **eICId\_BinInPin** | Дискретные входы датчиков 24В и 220В, состояние реле РПВ) | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 50 | 100 | Нет контроля |
| **eICId\_U\_GENrms** | Выходное напряжение контрольного генератора (среднеквадратическое значение) | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckOff – нет контроля | 50 | 300 | Нет контроля |
| **eICId\_U\_R1rms** | Напряжение датчика контроля положения стрелки 1 (среднеквадратическое значение) | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckDeltaP – абсолютная разность находится в пределах «deltaP» (процент) | 50 | 300 | Нет контроля |
| **eICId\_U\_R2rms** | Напряжение датчика контроля положения стрелки 2 (среднеквадратическое значение) | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckDeltaP – абсолютная разность находится в пределах «deltaP» (процент) | 50 | 300 | Нет контроля |
| **eICId\_IV\_value** | Мгновенное значение тока фазы V | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckDeltaP – абсолютная разность находится в пределах «deltaP» (процент) | 50 | 300 | Нет контроля |
| **eICId\_IU\_value** | Мгновенное значение тока фазы U | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckDeltaP – абсолютная разность находится в пределах «deltaP» (процент) | 50 | 300 | Нет контроля |
| **eICId\_IW\_value** | Мгновенное значение тока фазы W | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckDeltaP –абсолютная разность находится в пределах «deltaP» (процент) | 50 | 300 | Нет контроля |
| **eICId\_IV\_rms** | Среднеквадратическое значение тока фазы V | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckDeltaX – абсолютная разность находится в пределах «deltaХ» (значение) | 50 | 300 | Нет контроля |
| **eICId\_IU\_rms** | Среднеквадратическое значение тока фазы U | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckDeltaX – абсолютная разность находится в пределах «deltaХ» (значение) | 50 | 300 | Нет контроля |
| **eICId\_IW\_rms** | Среднеквадратическое значение тока фазы W | eScriptSync – синхронизация параметра | eProcSyncAverage – выбор арифметически среднего значения | eProcCheckDeltaX– абсолютная разность находится в пределах «deltaХ» (значение) | 50 | 300 | Нет контроля |
| **eICId\_ControlMK** | Обобщенные результаты проверок МК | eScriptSync – синхронизация параметра | eProcSyncEqual – обновление, если совпадают | eProcCheckEqual – абсолютное совпадение данных | 10 | 20 | 128 |

1. Исключения
   1. Обработка исключений

В процессе работы программы, могут возникать критические ошибки такие как: неисправность аппаратуры, неисправность МК, сбои в работе алгоритма программы, которые могут привести к опасным отказам. В каждом программном компоненте должен выполнятся анализ подобных ошибок и, в случае их обнаружения, должен выполнятся переход в ЗС при помощи группы макроопределений ASSERT из модулей asserts.h и asserts\_ex.h.

Макроопределения этой группы проверяют условие на истинность и, если условие оказалось ложью, вызывают функцию sysAssertException, которая инициирует переход в ЗС. Функция sysAssertException может работать в режиме отладки (см. 6.1.3).

Все исключения должны иметь свой уникальный идентификатор, который состоит из двух частей: идентификатора группы исключения и идентификатора исключения. Группа исключений определяет тип объекта, вызвавшего исключение, а идентификатор исключения определяет код исключения в группе. Группы и идентификаторы исключений определяются в файле ProtectionState\_codes.h.

Группа макроопределений ASSERT состоит из следующих макроопределений:

* ERROR\_ID(group\_, id\_);
* ERROR();
* ASSERT\_ID(group\_, id\_, test\_);
* ASSERT(test\_);
* CASSERT\_ID(group\_, id\_, counter\_, maxcount\_, test\_);
* CASSERT(counter\_, maxcount\_, test\_);
* ERROR\_EX\_ID(group\_, id\_ , p1\_, p2\_, p3\_, p4\_);
* ERROR\_EX(p1\_, p2\_, p3\_, p4\_);
* ASSERT\_EX\_ID(group\_, id\_, test\_ , p1\_, p2\_, p3\_, p4\_);
* ASSERT\_EX\_(test\_ , p1\_, p2\_, p3\_, p4\_);
* CASSERT\_EX\_ID(group\_, id\_, counter\_, maxcount\_, test\_ , p1\_, p2\_, p3\_, p4\_);
* CASSERT\_EX\_(counter\_, maxcount\_, test\_ , p1\_, p2\_, p3\_, p4\_).

Параметры вызова этих макроопределений:

* group\_ ‒ группа исключения;
* id\_ ‒ идентификатор исключения;
* test\_ ‒ проверяемое условие;
* counter\_ ‒ счетчик для организации задержки перед переходом в ЗС;
* maxcount\_ ‒ максимально допустимое значение счетчика задержки;
* p1\_, p2\_, p3\_, p4\_ ‒ параметры отказа, тип uint16\_t.

Макроопределение ERROR\_ID принимает идентификатор группы и исключения без условия и всегда приводит к вызову функции sysAssertException. Каждое макроопределение имеет два «двойника»: обработка исключений в режиме отладки и обработка исключений со счетчиком.

Макроопределение ASSERT\_ID в качестве параметров принимает идентификатор группы исключений, идентификатор исключения в группе и проверяемое условие. Если проверяемое условие не выполняется, вызывается функция sysAssertException.

Макроопределение CASSERT\_ID в качестве параметров принимает идентификатор группы исключений, идентификатор исключения в группе, счетчик ошибок, допустимое количество ошибок и проверяемое условие. Если проверяемое условие не выполняется, тогда значение счетчика увеличивается на величину CASSERT\_INC. Если увеличенное значение счетчика превысило максимально допустимое количество ошибок, вызывается функция sysAssertException. Если же проверяемое условие выполняется, значение счетчика уменьшается на величину CASSERT\_DEC до нуля.

Макроопределения без окончания «\_ID» вызывают исключения со значениями группы и идентификатора исключения равным ASSERT\_DEF\_CODE.

* + 1. Обработка отладочных исключений

Каждое макроопределение (раздел 6) имеет копию, название которой начинается с префикса «D»:

* DERROR\_ID(group\_, id\_);
* DERROR();
* DASSERT\_ID(group\_, id\_, test\_);
* DASSERT(test\_);
* DCASSERT\_ID(group\_, id\_, counter\_, maxcount\_, test\_);
* DCASSERT(counter\_, maxcount\_, test\_);
* DERROR\_EX\_ID(group\_, id\_ , p1\_, p2\_, p3\_, p4\_);
* DERROR\_EX(p1\_, p2\_, p3\_, p4\_);
* DASSERT\_EX\_ID(group\_, id\_, test\_ , p1\_, p2\_, p3\_, p4\_);
* DASSERT\_EX\_(test\_ , p1\_, p2\_, p3\_, p4\_);
* DCASSERT\_EX\_ID(group\_, id\_, counter\_, maxcount\_, test\_ , p1\_, p2\_, p3\_, p4\_);
* DCASSERT\_EX\_(counter\_, maxcount\_, test\_ , p1\_, p2\_, p3\_, p4\_).

Такие макроопределения вызывают исключения только в режиме отладки, когда определена директива ENABLED\_DEBUG\_ASSERT.

* + 1. Отключение обработки исключений

Для того, чтобы отключить вызов исключений, необходимо определить директиву DISABLED\_ASSERT.

* + 1. Отладочный режим функции sysAssertException

При определении директивы DEBUG\_EXCEPTION включается отладочный режим работы функции sysAssertException.

В этом режиме функция не вызывает функцию перехода в ЗС, а при помощи макроопределений из состава UNITY вызывает ошибку теста, с выводом информации о месте, где было вызвано исключение.

В отладочном режиме доступны две глобальные переменные exceptionExpectedErrorCode и exceptionErrorCodeIsExpected. Эти переменные используются при построении тестов, успех которых заключается в вызове конкретного исключения. Для того, чтобы тест при вызове исключения не был забракован, необходимо переменной exceptionExpectedErrorCode присвоить значение ожидаемого исключения. Значение ожидаемого исключения формируется из номера группы исключений и идентификатора исключения при помощи макроопределения SET\_CODE\_ID(group\_, id\_). Если произойдет вызов указанного исключения, функция sysAssertException не забракует тест, а установит глобальную переменную exceptionErrorCodeIsExpected в значение, отличное от нуля.